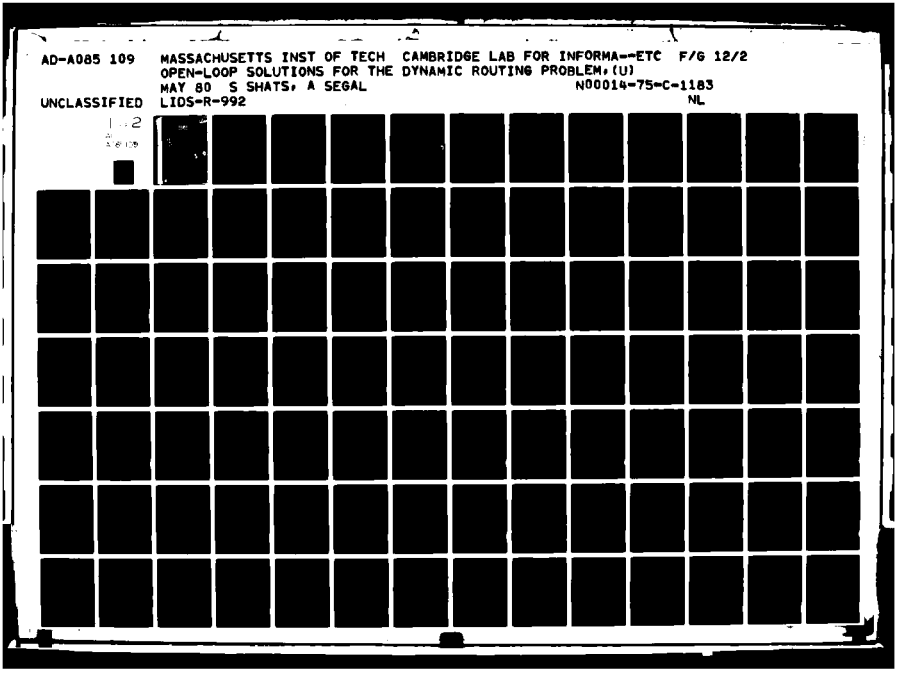
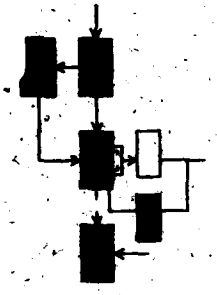
AD-A085 109 MASSACHUSETTS INST OF TECH CAMBRIDGE LAB FOR INFORMA--ETC F/G 12/2
OPEN-LOOP SOLUTIONS FOR THE DYNAMIC ROUTING PROBLEM,(U)
MAY 80 S SHATS, A SEGAL N00014-75-C-1183
UNCLASSIFIED LIDS-R-992 NL



May, 1980 LIDS-R-998

LEVEL II

Research Supported By:
AFPA Contract N00014-75-C-1183
OSP Number 82553
ONR Contract N00014-77-C-0532
OSP Number 83552



OPEN-LOOP SOLUTIONS FOR THE DYNAMIC ROUTING PROBLEM

Samuel Shats
Adrian Segall

Laboratory for Information and Decision Systems
Formerly
Electronic Systems Laboratory
MASSACHUSETTS INSTITUTE OF TECHNOLOGY, CAMBRIDGE, MASSACHUSETTS 021

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER
2. GOVT ACCESSION NO. AD-A085109
3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
(6) Open-Loop Solutions for the Dynamic Routing Problem

5. TYPE OF REPORT & PERIOD COVERED

6. PERFORMING ORG. REPORT NUMBER
(14) LIDS-R-992

7. AUTHOR(s)
(10) Samuel/Shats
Adrian/Segall

8. CONTRACT OR GRANT NUMBER(s)
(15) Contract N00014-75-C-1183,
N00014-77-C-0532

9. PERFORMING ORGANIZATION NAME AND ADDRESS
MIT
Laboratory for Information and Decision Systems
Cambridge, MA02139

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
Program Code No. 5T10
ONR Identifying No. 049-383

11. CONTROLLING OFFICE NAME AND ADDRESS
(11) Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, Virginia 22209

12. REPORT DATE
May 1980

13. NUMBER OF PAGES
141 pages

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)
Office of Naval Research
Information Systems Program
Code 437
Arlington, VA 22217
(12) 149

15. SECURITY CLASS. (of this report)
Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This work deals with the problem of obtaining an open-loop solution to the minimum delay dynamic routing problem.

The dynamic routing problem is stated using a dynamic model suggested in previous works. This work uses some previously known properties of the optimal solution and formulates the routing problem as a cubic optimization problem. In general such problems are very hard to solve; however, the specific problem at hand is finally formulated as a nonconvex quadratic program

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

410950

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

by using its special structure.

Two different approaches, based on the latter representation of the problem, are proposed:

(a) Utilization of existing methods for solving nonconvex quadratic programs,

(b) Development of a special purpose allgorithm.

The algorithm is developed for single destination networks with unity weightings in the cost functional, and it finds the optimal solution by solving a series of linear programs.

The algorithm is based on a series of specially developed theorems. These theorems provide us with new insight into the behaviour of the dynamic routing in networks.

The meathod is implemented by a computer program and several examples are run to test its applicability.

INSTRUCTIONS FOR PREPARATION OF REPORT DOCUMENTATION PAGE

RESPONSIBILITY. The controlling DoD office will be responsible for completion of the Report Documentation Page, DD Form 1473, in all technical reports prepared by or for DoD organizations.

CLASSIFICATION. Since this Report Documentation Page, DD Form 1473, is used in preparing announcements, bibliographies, and data banks, it should be unclassified if possible. If a classification is required, identify the classified items on the page by the appropriate symbol.

COMPLETION GUIDE

General. Make Blocks 1, 4, 5, 6, 7, 11, 13, 15, and 16 agree with the corresponding information on the report cover. Leave Blocks 2 and 3 blank.

Block 1. Report Number. Enter the unique alphanumeric report number shown on the cover.

Block 2. Government Accession No. Leave Blank. This space is for use by the Defense Documentation Center.

Block 3. Recipient's Catalog Number. Leave blank. This space is for the use of the report recipient to assist in future retrieval of the document.

Block 4. Title and Subtitle. Enter the title in all capital letters exactly as it appears on the publication. Titles should be unclassified whenever possible. Write out the English equivalent for Greek letters and mathematical symbols in the title *(see "Abstracting Scientific and Technical Reports of Defense-sponsored RDT/E," AD-667 000)*. If the report has a subtitle, this subtitle should follow the main title, be separated by a comma or semicolon if appropriate, and be initially capitalized. If a publication has a title in a foreign language, translate the title into English and follow the English translation with the title in the original language. Make every effort to simplify the title before publication.

Block 5. Type of Report and Period Covered. Indicate here whether report is interim, final, etc., and, if applicable, inclusive dates of period covered, such as the life of a contract covered in a final contractor report.

Block 6. Performing Organization Report Number. Only numbers other than the official report number shown in Block 1, such as series numbers for in-house reports or a contractor/grantee number assigned by him, will be placed in this space. If no such numbers are used, leave this space blank.

Block 7. Author*(s)*. Include corresponding information from the report cover. Give the name*(s)* of the author*(s)* in conventional order *(for example, John R. Doe or, if author prefers, J. Robert Doe)*. In addition, list the affiliation of an author if it differs from that of the performing organization.

Block 8. Contract or Grant Number*(s)*. For a contractor or grantee report, enter the complete contract or grant number*(s)* under which the work reported was accomplished. Leave blank in in-house reports.

Block 9. Performing Organization Name and Address. For in-house reports enter the name and address, including office symbol, of the performing activity. For contractor or grantee reports enter the name and address of the contractor or grantee who prepared the report and identify the appropriate corporate division, school, laboratory, etc., of the author. List city, state, and ZIP Code.

Block 10. Program Element, Project, Task Area, and Work Unit Numbers. Enter here the number code from the applicable Department of Defense form, such as the DD Form 1498, "Research and Technology Work Unit Summary" or the DD Form 1634, "Research and Development Planning Summary," which identifies the program element, project, task area, and work unit or equivalent under which the work was authorized.

Block 11. Controlling Office Name and Address. Enter the full, official name and address, including office symbol, of the controlling office. *(Equates to funding/sponsoring agency. For definition see DoD Directive 5200.20, "Distribution Statements on Technical Documents.")*

Block 12. Report Date. Enter here the day, month, and year or month and year as shown on the cover.

Block 13. Number of Pages. Enter the total number of pages.

Block 14. Monitoring Agency Name and Address *(if different from Controlling Office)*. For use when the controlling or funding office does not directly administer a project, contract, or grant, but delegates the administrative responsibility to another organization.

Blocks 15 & 15a. Security Classification of the Report: Declassification/Downgrading Schedule of the Report. Enter in 15 the highest classification of the report. If appropriate, enter in 15a the declassification/downgrading schedule of the report, using the abbreviations for declassification/downgrading schedules listed in paragraph 4-207 of DoD 5200.1-R.

Block 16. Distribution Statement of the Report. Insert here the applicable distribution statement of the report from DoD Directive 5200.20, "Distribution Statements on Technical Documents."

Block 17. Distribution Statement *(of the abstract entered in Block 20, if different from the distribution statement of the report)*. Insert here the applicable distribution statement of the abstract from DoD Directive 5200.20, "Distribution Statements on Technical Documents."

Block 18. Supplementary Notes. Enter information not included elsewhere but useful, such as: Prepared in cooperation with . . . Translation of *(or by)* . . . Presented at *conference of* . . . To be published in . . .

Block 19. Key Words. Select terms or short phrases that identify the principal subjects covered in the report, and are sufficiently specific and precise to be used as index entries for cataloging, conforming to standard terminology. The DoD "Thesaurus of Engineering and Scientific Terms" *(TEST)*, AD-672 000, can be helpful.

Block 20. Abstract. The abstract should be a brief *(not to exceed 200 words)* factual summary of the most significant information contained in the report. If possible, the abstract of a classified report should be unclassified and the abstract to an unclassified report should consist of publicly- releasable information. If the report contains a significant bibliography or literature survey, mention it here. For information on preparing abstracts see "Abstracting Scientific and Technical Reports of Defense-Sponsored RDT&E," AD-667 000.

☆ U.S. GOVERNMENT PRINTING OFFICE : 1973—729-091/1431 3-1

May, 1980 LIDS-R-992

OPEN-LOOP SOLUTIONS

FOR THE DYNAMIC ROUTING PROBLEM

Samuel Shats and Adrian Segall

Department of Electrical Engineering
Technion - Israel Institute of Technology
Haifa, Israel

The work of A. Segall was performed on a consulting agreement with the Laboratory for Information and Decision Systems at M.I.T., Cambridge, Mass., and was supported in part by the Advanced Research Project Agency of the U.S. Department of Defense (monitored by ONR) under Contract No. N00014-75-C-1183, and in part by the Office of Naval Research under Contract No. ONR/N00014-77-C-0532.

Table of Contents

Abstract

This work deals with the problem of obtaining an open-loop solution to the minimum delay dynamic routing problem.

The dynamic routing problem is stated using a dynamic model suggested in previous works. This work uses some previously known properties of the optimal solution and formulates the routing problem as a cubic optimization problem. In general such problems are very hard to solve; however, the specific problem at hand is finally formulated as a nonconvex quadratic program by using its special structure.

Two different approaches, based on the latter representation of the problem, are proposed:

(a) Utilization of existing methods for solving nonconvex quadratic programs.

(b) Development of a special purpose algorithm.

The algorithm is developed for single destination networks with unity weightings in the cost functional, and it finds the optimal solution by solving a series of linear programs.

The algorithm is based on a series of specially developed theorems. These theorems provide us with new insight into the behaviour of the dynamic routing in networks.

The method is implemented by a computer program and several examples are run to test its applicability.

Glossary of Notations

N - set of network nodes

$\bar{N}$ - set of network nodes *not* including the destination node

L - set of network links

(i,k) - directed link from node i to node k

C_{ik} - capacity of link (i,k)

$\underline{x}$ - vector of state variables

X - sum of the elements of vector $\underline{x}$

$\underline{u}$ - vector of control variables

$\underline{u}_i$ - vector of control variables on $[t_{i-1}, t_i)$

I_p - set of states travelling on interior arcs on $[t_{p-1}, t_p)$

B_p - set of states travelling on boundary arcs on $[t_{p-1}, t_p)$

d - destination node

F - set of numbers $\{1,2,\ldots,f\}$

t_i - switching time

τ_i - time period from t_{i-1} to t_i

$\underline{z}_i$ - vector of variables equal to $\underline{u}_i \tau_i$

Z - plane with axes $\underline{\alpha}^T \underline{x}$ and t

$(\cdot)^*$ - variable referring to the optimal solution

$E(i)$ - collection of nodes k such that $(i,k) \in L$

$I(i)$ - collection of nodes ℓ such that $(\ell,i) \in L$

Chapter 1

Introduction

1.1 Basic Concepts

This work considers the problem of dynamic routing in data-communication networks, in the sense that the routing depends on the instantaneous queue length at the network nodes. A dynamic model for this problem has been proposed in [G7] and closed-loop feedback optimal solutions for this model have been investigated in [G7], [G4], [G5], [G6] and [G2]. It has been shown that the full closed-loop solution requires the construction of polyhedral convex cones, named "feedback control regions", with the property that the optimal control set is constant within each region.

In this work we present a different approach whereby we seek the open-loop optimal control for the dynamic model of [G7]. In this case the controls depend only on the initial condition (i.e., the initial congestion of the network).

In this work we will propose some alternatives for solving this problem, all of them based on a change of variables (proposed by F. Moss) in Segall's dynamic model. (For other approaches to the open-loop problem see [G4], pp. 45-49).

1.2 The Dynamic Model of a Data Communication Network [G7]

1.2.1 Topological Representation

A data communication network may be represented as a set of nodes interconnected by a number of links where to each node are connected a

number of "users" (see Figure 1.1). The links are communication channels used for the transmission of information from node to node, which are assumed to carry messages in only one direction.

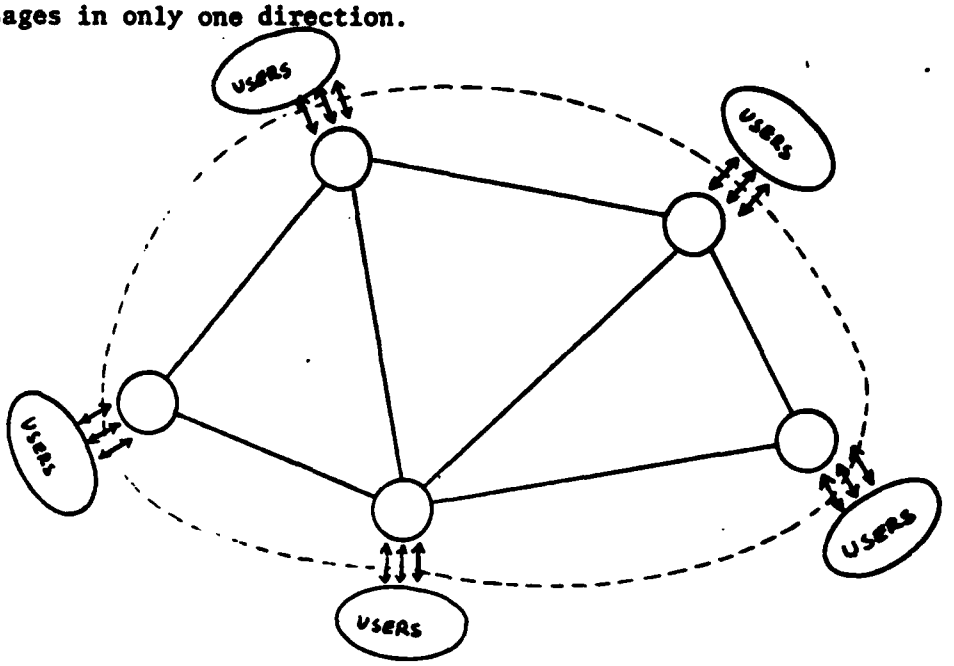


Figure 1.1: Data communication network.

In a network consisting of $n+1$ nodes we associate with each node an integer in the set:

$$N = \{1,2,\ldots,n,n+1\} \quad .$$

We now denote (see Figure 1.2):

1. (i,k) is the link connecting node i to node k.
2. $L \triangleq \{(i,k)$ such that $i,k \in N$ and there exists a link from i to $k\}$
   that is, L is the collection of all links in the network.
3. $C_{ik} \triangleq \{$capacity of link (i,k) in units of traffic/unit time, $(i,k) \in L\}$.
4. $E(i) \triangleq \{$collection of nodes k such that $(i,k) \in L$, $\forall i \in N\}$.
5. $I(i) \triangleq \{$collection of nodes ℓ such that $(\ell,i) \in L$, $\forall i \in N\}$.

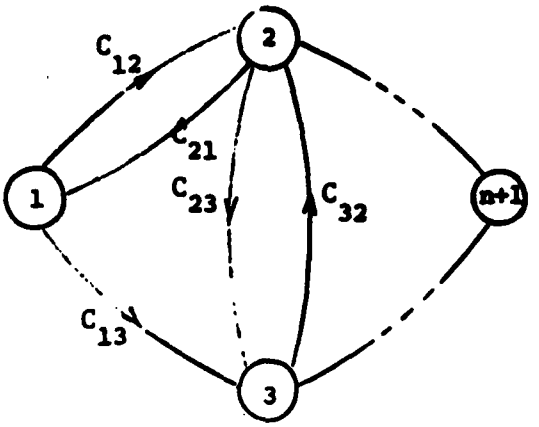


Figure 1.2: Topology of a data communication network.

1.2.2 Input Flow, State and Control Variables

The users at each node in the network may input messages whose destination is any of the other nodes in the network. We denote this flow input to the network by:

$$a_i^j(t) \triangleq \{\text{rate of traffic with destination } j \text{ arriving at node } i \text{ (from associated users) at time } t\}.$$

A particular message may pass through a number of nodes before arriving at the destination node. Since the capacity of the links is finite, every message that arrives at a node must wait in a queue for being forwarded. Once a message reaches its destination node, it is immediately forwarded to the appropriate user without further storage. The principal source of delay in the transmission of information in the network is the queueing delay in the nodes.

For each node we define the following state variables:

$$x_i^j(t) \triangleq \{\text{amount of traffic at node } i \text{ at time } t \text{ whose final destination is node } j, \text{ where } i,j \in N,\ i \neq j\}.$$

Finally, we will define the control variables of the state space model as:

$$u_{ik}^{j} \triangleq \{\text{flow of information on the link } (i,k) \text{ at time } t \text{ with final destination } j\} .$$

1.2.3 Dynamic Equations and Constraints

Assuming that the inputs are deterministic functions of time, the time rate of change of the number of messages contained in each node is given by

$$\dot{x}_i^j(t) = a_i^j(t) - \sum_{k\in E(i)} u_{ik}^j(t) + \sum_{\substack{\ell\in I(i)\\ \ell\neq j}} u_{\ell i}^j(t) \quad , \quad \forall\, i,j \in N, \quad j \neq i \quad , \tag{1.1}$$

with the control and state constraints:

$$x_i^j(t) \geq 0 \quad , \quad \forall\, t \quad , \tag{1.2}$$

$$u_{ik}^j(t) \geq 0 \quad , \quad \forall\, t \quad , \tag{1.3}$$

$$\sum_{j\in N} u_{ik}^j(t) \leq C_{ik} \quad , \quad \forall\, t \quad , \quad \forall\, (i,k) \in L \quad , \quad j = i \tag{1.4}$$

where C_{ik} is the capacity of the link (i,k) .

1.2.4 Performance Index

As indicated in Section 1.2.2, $x(t)$ is the amount of message residing in some node at time t, then the quantity

$$\int_{t_o}^{t_f} x(t)dt \tag{1.5}$$

is the total time spent in this node by the traffic passing through it during $[t_o,t_f]$, when $x(t_f) = 0$. That is, expression (1.5) is the total delay in

the node experienced by the messages represented by $x(t)$. Hence, the total delay experienced by all the messages as they travel through the network during $[t_o, t_f]$ is given by

$$D = \int_{t_o}^{t_f} [\sum_{i,j} x_i^j(t)]dt \ , \quad i,j \in N, \ j \neq i \ , \tag{1.6}$$

where t_f is defined as the time at which all the message storage state variables x_i^j go to zero. Priorities can be accommodated in the cost functional (1.6) by associating non-equal weightings α_i^j to the appropriate state variables, so that we have

$$J = \int_{t_o}^{t_f} [\sum_{i,j} \alpha_i^j x_i^j(t)]dt \ , \quad i,j \in N, \ j \neq i \ , \tag{1.7}$$

with t_f defined as above.

1.3 Research Overview

1.3.1 Objective and Approach of the Research

The goal of this research is to obtain an algorithm for solving the open-loop optimal dynamic routing problem in data communication networks.

For a given initial condition, the open-loop solution will give the optimal control that takes the initial state to zero, so that the total delay is minimized. This is in contradiction to the feedback (closed-loop) solution, where the optimal control is to be found for every value of the state.

The reason for looking for an open-loop solution is twofold. First, it turns out from [G4] and [G2] that a closed-loop solution requires a large number of calculations and of memory space for networks of reasonable size.

Second, there are still a number of unsolved problems related to the representation of the closed-loop solution, such as determination of the different regions in the state space, determination of the region the initial condition belongs to, and determination of the time in which the state trajectory reaches another region for knowing when to change the controls.

The approach used in this research was to make use of known results from previous works for receiving a new formulation of the dynamic routing problem and then by means of a change of variables transform it into a quadratic program. This quadratic problem was solved by using linear programs after developing an appropriate theoretical background. Likewise, other approaches for solving it, based on known methods for solving quadratic programs are proposed.

1.3.2 Synopsis of the Research

The purpose of this section is to provide a brief summary of the remaining chapters.

Chapter 2: In this chapter the optimal control problem is defined and a number of results relating to the feedback solution, as appear in [G4], [G5], [G6] and [G7], are presented. Then, a summary of the properties of single destination networks with unity weightings in the cost functional is presented.

Chapter 3: In this chapter the optimal control problem defined in Chapter 2 is presented as a cubic optimization problem and after a change of variables as a quadratic program. Then it is demonstrated that the cost function is not quasi-convex, pseudo-convex, concave or quasi-concave. These properties are explained in Appendix A.

Chapter 4: In this chapter special properties of single-destination networks with unity weightings in the cost functional are used for developing a theory that enables us to construct an algorithm for solving the open loop optimal routing problem for this kind of networks by using linear programs. The proposed algorithm is implemented on a computer in Appendix C.

Chapter 5: In this chapter a number of alternative approaches for solving the problem are proposed, all of them based on existing techniques for solving the nonconvex quadratic programming. These techniques are overviewed in Appendix B.

Chapter 6: In this last chapter the results of the research are analyzed and some ideas for further research are proposed.

1.3.3 Contributions

The contributions of this work are:

(a) For the first time the open-loop dynamic routing problem is investigated.

(b) As previously said in Section 1.2, the variables of the original formulation of the optimal dynamic routing problem are the state and control variables. In this work we made use of the fact that the optimal solution is of the bang-bang type in order to formulate the open-loop routing problem as dependent on new variables: the controls and the time these controls are applied. This new formulation is a cubic optimization problem with a nonconvex constraint region. This type of problems is quite difficult, but the special form of the cost function and of the constraint region permits us to formulate the problem as a quadratic program.

(c) The properties of this quadratic program are investigated, and it is shown that the cost function does not have any of the following properties: convexity, quasi-convexity, pseudo-convexity, concavity or quasi-concavity.

(d) Since the cost function does not possess any special property that will allow to solve the open-loop dynamic routing problem easily by existing methods, two basic approaches for its solution are proposed. The first is based on existing methods from nonconvex quadratic programming, and the second consists of developing a special algorithm.

(e) The existing methods for solving nonconvex quadratic programs are reviewed, and the methods that seem to be most adequate for the solution of our problem are more deeply analyzed. All results are condensed in an Appendix that may provide the reader interested in nonconvex quadratic programming with important first information, of existing methods and with an extensive bibliography on the subject.

(f) The main part of this work consists of the development of an algorithm that reduces the open-loop dynamic routing problem for single destination networks with unity weightings in the cost functional to a series of linear programs. This algorithm is developed based on a number of theorems that provide the required theoretical background. The algorithm consists of first finding the final time (i.e. the time the network empties) for the optimal solution and the non-switch-optimal-solution (i.e. the optimal solution among all the solutions having no switches). From the theoretical background follows that this can be calculated by solving a small linear program. The fact that the final time is known enables us to find the one-switch-optimal-solution by solving another linear program. The

algorithm continues adding switches in each of the intervals between switches. Each new step is solved by linear programming. The algorithm will stop when the addition of a series of new possible switches does not decrease the value of the optimal cost.

(g) The only limitation on the capability of the algorithm to solve large networks are the properties of the linear program used. In this work we use a linear program from an existing library in order to test the algorithm.

Chapter 2

The Optimal Control Problem and its Solution

2.1 Introduction

This chapter deals with the fundamental laws governing the solution of the optimal dynamic routing problem in data communication networks.

Based on the model described in Section 1.2, we present the optimal control problem, the necessary and sufficient condition for its solution and the properties of the optimal control. Then, in Section 2.5 we present special properties of single destination networks with unity weightings in the cost functional.

In this work we deal only with networks without input flow, that is:

$$a_i^j(t) = 0 \ , \ \forall i,j,t \ .$$

All results in this chapter are presented without proof. The proofs appear in [G2], [G4], [G5] and [G6].

2.2 The Optimal Control Problem

To facilitate the expression of the problem in mathematical form, we define the column vectors $\underline{x}$ and $\underline{\alpha}$ (dimension n), $\underline{u}$ (dimension m, with m being the number of controls to be determined), and $\underline{C}$ (dimension r, with r being the number of links in the network), which are respectively concatenations of the state variables, cost functional weightings, control variables and link capacities. For a given network topology we define the $n \times m$ incidence matrix B as follows: associated with every

state variable x_i^j, there is a row $\underline{b}^T$ of B such that

$$\underline{b}^T\underline{u} = -\sum_{k\in E(i)} u_{ik}^j + \sum_{\ell\in I(i)} u_{\ell i}^j \quad , \quad i,j \in N, \quad j \neq i \ . \tag{2.1}$$

B is composed entirely of +1's, -1's and 0's, and every column of B has at most two non-zero elements. If a particular column has only one non-zero entry then it is -1.

In a similar way, we define the $r \times m$ matrix D: associated with every link (i,k) is a row $\underline{d}^T$ of D such that

$$\underline{d}^T\underline{u} \leq C_{ik} \quad , \tag{2.2}$$

represents the constraint (1.4). The elements of D are 0's and 1's only, and each column has precisely one +1.

Based on the above notations we can state the data communication network dynamic message routing problem as follows:

Problem 2.1

Find the set of controls $\underline{u}$ as a function of time (and state, in the case of closed loop solution)

$$u(t) \ , \quad t \in [t_o, t_f] \ , \tag{2.3}$$

that will bring the state from a given initial condition $\underline{x}(t_o) = \underline{x}_o$ to $\underline{x}(t_f) = \underline{0}$ and will minimize the cost functional

$$J = \int_{t_o}^{t_f} [\underline{\alpha}^T\underline{x}(t)]dt \quad , \tag{2.4}$$

subject to the state dynamics

$$\dot{\underline{x}}(t) = B\underline{u}(t) \quad , \tag{2.5}$$

and constraints on the state and control variables

$$\underline{x}(t) \geq 0 \ , \quad \forall t \in [t_o, t_f] \quad , \tag{2.6}$$

$$\underline{u} \begin{cases} D\underline{u} \leq \underline{C} \\ \\ \underline{u} \geq 0 \end{cases} \tag{2.7}$$

□ Problem 2.1

Note that (2.3) - (2.7) define an optimal control problem with linear constraints and linear cost functional.

2.3 Necessary and Sufficiency Conditions

Theorem 2.1: Necessary Conditions (see [G5], pp. 13-19)

Let the scalar functional h be defined as follows:

$$h[\underline{u}(t), \underline{\lambda}(t)] \triangleq \underline{\lambda}^T(t)\dot{\underline{x}}(t) = \underline{\lambda}^T(t)B\underline{u}(t) \ . \tag{2.8}$$

A necessary condition for the control law $\underline{u}^*(\cdot) \in \mathcal{U}$ to be optimal for Problem 2.1 is that it minimize h pointwise in time, namely

$$\underline{\lambda}^T(t)B\underline{u}^*(t) \leq \underline{\lambda}^T(t)B\underline{u}(t) \tag{2.9}$$

$$\forall \underline{u}(t) \in \mathcal{U} \qquad t \in [t_o, t_f] \ .$$

The costate $\underline{\lambda}(t)$ is possibly a discontinuous function which satisfies the following differential equation

$$-d\underline{\lambda}(t) = \underline{a}dt + d\underline{\eta}(t) \ , \quad t \in [t_o, t_f] \ , \tag{2.10}$$

where componentwise $d\underline{\eta}(t)$ satisfies the following complementary slackness

condition

$$\left.\begin{aligned} x_i^j(t)\,d\eta_i^j(t) &= 0 \\ d\eta_i^j(t) &\leq 0 \end{aligned}\right\} \quad \forall t \in [t_o, t_f] \quad i,j \in N, \quad j \neq i \ . \tag{2.11, 2.12}$$

The terminal boundary condition for the costate differential equation is

$$\underline{\lambda}(t_f) = \underline{\nu} \quad \text{free} \quad , \tag{2.13}$$

and the transversality condition is

$$\underline{\lambda}^T(t_f)\dot{\underline{x}}(t_f) = \underline{0} \quad . \tag{2.14}$$

Finally, the function h is everywhere continuous, i.e.

$$h[\underline{u}(t^-), \underline{\lambda}(t^-)] = h[\underline{u}(t^+), \underline{\lambda}(t^+)] \ , \quad \forall t \in [t_o, t_f] \ . \tag{2.15}$$

□ Theorem 2.1

The costate trajectories depend on the value of the corresponding state variables. When the state $x_i^j > 0$, equation (2.11) implies that $d\eta_i^j = 0$; therefore by differentiating equation (2.1) with respect to time we obtain

$$-\dot{\lambda}_i^j(t) = \alpha_i^j \quad . \tag{2.16}$$

When the state $x_i^j = 0$, its respective costate is possible discontinuous, depending on the nature of η_i^j. At points for which η_i^j is absolutely continuous we define $\mu_i^j \triangleq \dfrac{d\eta_i^j(t)}{dt}$ and by differentiating (2.10) with respect to time we have, based also on (2.11) and (2.12), that

$$-\dot{\lambda}_i^j(t) = \alpha_i^j + \mu_i^j(t) \ , \quad \mu_i^j(t) \leq 0 \ . \tag{2.17}$$

On the other hand, at times when η_i^j experience jumps of magnitude $\Delta\eta_i^j$

(see (2.10))

$$\Delta\lambda_i^j = -\Delta\eta_i^j \quad , \tag{2.18}$$

and from equations (2.11) and (2.12) it is known that those jumps are positive. Hence

$$\Delta\lambda_i \geq 0 \quad . \tag{2.19}$$

Theorem 2.2: Sufficiency Conditions (see [GS], pp. 19-20)

The necessary conditions of Theorem 2.1 are sufficient.

□ Theorem 2.2

We conclude from Theorem 2.2 that all trajectories from $\underline{x}_o$ to $\underline{x}(t_f) = \underline{0}$ satisfying the necessary conditions of Theorem 2.1 are optimal.

2.4 Characterization of the Optimal Control

From the inequality (2.9) we see that the optimal control $\underline{u}^*(\cdot)$ is received by solving at every time $\tau \in [t_o, t_f]$ the linear program:

$$\underline{u}^*(\tau) = \text{ARG} \underset{\underline{u}(\tau)\in U}{\text{MIN}} [\underline{\lambda}^T(\tau)B\underline{u}(\tau)] \quad . \tag{2.20}$$

The minimization can be performed on one link at a time. Consider the link (i,k) and a possible set of associated controls

$$u_{ik}^1, u_{ik}^2, \ldots, u_{ik}^{i-1}, u_{ik}^{i+1}, \ldots, u_{ik}^{n+1} \quad .$$

A given control may appear in one of the following ways:

1. u_{ik}^j enters into exactly two state equations (see (2.5)):

$$\dot{x}_i^j(t) = -u_{ik}^j(t) - \sum_{\substack{q\in E(i)\\ q\neq k}} u_{iq}^j(t) + \sum_{\ell\in E(i)} u_{\ell i}^j(t) \quad , \tag{2.21a}$$

$$\dot{x}_k^j(t) = u_{ik}^j(t) + \sum_{\substack{\ell\in I(k)\\ \ell\neq i}} u_{\ell k}^j(t) - \sum_{q\in E(k)} u_{kq}^j(t) \quad . \tag{2.21b}$$

2. u_{ik}^{k} enters into exactly one state equation:

$$\dot{x}_i^k(t) = -u_{ik}^k(t) - \sum_{\substack{q\in E(i)\\ q\neq k}} u_{iq}^j(t) + \sum_{\ell\in I(i)} u_{\ell i}^j(t) \quad . \tag{2.22}$$

Hence, all controls on link (i,k) contribute the following terms to $\underline{\lambda}^T B\underline{u}$:

$$(\lambda_k^1(t) - \lambda_i^1(t))u_{ik}^1(t) + (\lambda_k^2(t) - \lambda_i^2(t))u_{ik}^2(t) + \ldots +$$

$$(\lambda_k^{i-1}(t) - \lambda_i^{i-1}(t))u_{ik}^{i-1}(t) + (\lambda_k^{i+1}(t) - \lambda_i^{i+1}(t))u_{ik}^{i+1}(t) + \ldots +$$

$$(\lambda_k^{n+1}(t) - \lambda_i^{n+1}(t))u_{ik}^{n+1}(t) \quad , \tag{2.23}$$

where $\lambda_k^k(t) = 0$. Equations (1.4), (2.20) and (2.23) determine the <u>optimal control law at time t on (i,k)</u>:

1. $u_{ik}^{\ell} = C_{ik}$ and $u_{ik}^{j} = 0$, $\quad j \neq \ell$, (2.24)

if

$$(\lambda_k^{\ell}(t) - \lambda_i^{\ell}(t)) < (\lambda_k^j(t) - \lambda_i^j(t)) \quad , \quad \forall\, j \neq \ell \ ,$$

and

$$(\lambda_k^{\ell}(t) - \lambda_i^{\ell}(t)) < 0 \ .$$

2. $u_{ik}^{\ell}(t) + u_{ik}^{\ell+1}(t) + \ldots + u_{ik}^{m}(t) = C_{ik}$ (2.25)

$$u_{ik}^j(t) = 0 \quad , \quad \forall\, j \notin [\ell, \ell+1, \ldots, m]$$

if

$$(\lambda_k^{\ell}(t) - \lambda_i^{\ell}(t)) = (\lambda_k^{\ell+1}(t) - \lambda_i^{\ell+1}(t)) = \ldots =$$

$$= (\lambda_k^{m}(t) - \lambda_i^{m}(t)) < (\lambda_k^j(t) - \lambda_i^j(t)) \quad , \quad \forall\, j \notin [\ell, \ell+1, \ldots, m]$$

and

$$(\lambda_k^{\ell}(t) - \lambda_i^{\ell}(t)) = (\lambda_k^{\ell+1}(t) - \lambda_i^{\ell+1}(t)) = \ldots = (\lambda_k^{m}(t) - \lambda_i^{m}(t)) < 0 \ .$$

3. $$u_{ik}^{\ell}(t) + u_{ik}^{\ell+1}(t) + \ldots + u_{ik}^{m}(t) \leq C_{ik} \quad , \qquad (2.26)$$

$$u_{ik}^{j}(t) = 0 \quad , \quad j \notin [\ell, \ell+1, \ldots, m] \quad ,$$

if

$$(\lambda_k^{\ell}(t) - \lambda_i^{\ell}(t)) = (\lambda_k^{\ell+1}(t) - \lambda_i^{\ell+1}(t)) = \ldots =$$

$$= (\lambda_k^{m}(t) - \lambda_i^{m}(t)) = 0 \quad ,$$

and

$$(\lambda_k^{j}(t) - \lambda_i^{j}(t)) > 0 \quad , \quad j \notin [\ell, \ell+1, \ldots, m] \quad .$$

From the optimal control law above we conclude that:

Conclusion 2.1: (see [G4], pp. 72-73)

There always exists a solution to Problem 2.1 with controls piecewise constant in time, and trajectory with piecewise constant slopes.

□ Conclusion 2.1

From Conclusion 2.1 it is clear that there exists an optimal trajectory which can be characterized by a finite number of parameters. We now present a compact set of notations for specifying these parameters.

Let F be the set $[1,2,\ldots,f]$, and $t_o, t_1, \ldots, t_f$ be the switching times (see Figure 2.1), and then denote:

Definition 2.1

$$\underline{u}_i \triangleq \{\text{the control vector on } t \in [t_{i-1}, t_i)\} \quad , \quad i \in F \quad .$$

Definition 2.2

$$\tau_i \triangleq \{\text{the time period between } t_{i-1} \text{ and } t_i, \text{ i.e., } \tau_i = t_i - t_{i-1}\} \quad , \quad i \in F.$$

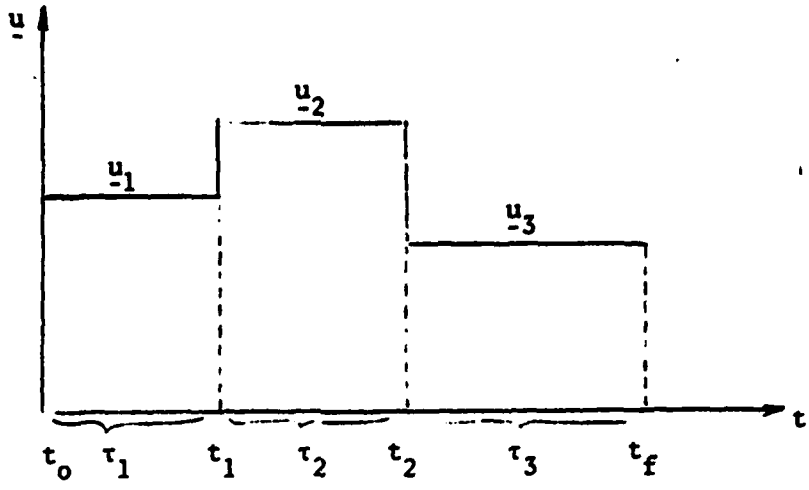


Figure 2.1: Illustration of $\underline{u}_i$ and τ_i .

Definition 2.3

$$B_p \triangleq \{x_i^j / x_i^j(t) = 0 \ , \quad \forall t \in [t_{p-1}, t_p)\} \ , \quad \forall p \in F.$$

Hence, B_p is the set of state variables that take value zero during the time interval $[t_{p-1}, t_p)$.

Definition 2.4

$$B(\underline{x}) = \{B_1, B_2, \ldots, B_f\} \ .$$

That is, $B(\underline{x})$ is the sequence of the sets B_p. $B(\underline{x})$ is called the boundary sequence.

Definition 2.5

$$I_p \triangleq \{x_i^j / x_i^j(t) > 0 \ , \quad \forall t \in [t_{p-1}, t_p)\} \ , \quad p \in F \ .$$

Hence, I_p is the set of state variables that are not at zero value during the time interval $[t_{p-1}, t_p)$.

2.5 Single Destination Networks with Unity Weightings in the Cost Functional

2.5.1 The Optimal Control Problem

Let us denote the destination node as the n+1 node from the set of nodes N and then define the set of nodes

$$\bar{N} = \{1,2,\ldots,n\} \quad ,$$

then, $\bar{N}$ is the set of all the nodes in the network besides the destination node. Since there is only one destination node, it is possible to omit the index that represents the destination node in the control and state variables (see Sections 1.2 and 2.2). Therefore we will write x_i instead of x_i^{n+1}, and u_{ik} instead of u_{ik}^{u+1}. Likewise, in each link (i,k) there is flow of information towards a single destination node; then, the D matrix (see (2.7)) is the identity matrix.

By using the above notation it is possible to state Problem 2.1 for single destination networks with unity weightings in the cost functional ($\underline{\alpha} = \underline{1}$) as follows:

Problem 2.2:

Find the set of controls $\underline{u}$ as a function of time (and state in the case of closed-loop solution)

$$\underline{u}(t) \quad , \quad t \in [t_o, t_f] \quad , \tag{2.27}$$

that will bring the state from a given initial condition $\underline{x}(t_o) = \underline{x}_o$ to $\underline{x}(t_f) = \underline{0}$ and will minimize the cost functional

$$J = \int_{t_o}^{\tau_f} [\sum_{i \in \bar{N}} \underline{x}_i] dt \quad , \tag{2.28}$$

subject to the state dynamics

$$\dot{\underline{x}}(t) = B\underline{u}(t) \quad , \tag{2.29}$$

and constraints on the state and control variables

$$\underline{x}(t) \geq 0 \quad , \quad t \in [t_o, t_f] \quad , \tag{2.30}$$

$$U\{0 \leq u_{ij} \leq C_{ij}\} \quad , \quad (i,j) \in L \tag{2.31}$$

□ Problem 2.2

2.5.2 Special Properties

Proposition 2.1 (see [G4], pp. 244-259, and [G2], pp. 77-83):

The costates $\lambda_i(t)$ are continuous functions of time and in addition $\lambda_i(t) \geq 0$, $i \in \bar{N}$, t.

□ Proposition 2.1

Proposition 2.2 (see [G4], pp. 263-267, and [G2], pp. 77-83):

There always exists an optimal solution satisfying $\dot{\underline{x}}(t) \leq 0$, $\forall t$.

□ Proposition 2.2

Proposition 2.3 (see [G2], pp. 51-53):

The set of all solutions to the problem:

$$\min_{x_i \in I_p} \sum \lambda_i \dot{x}_i \quad , \quad \lambda_i > 0 \quad , \quad x_i \in I_p \quad ,$$

satisfying

$$\dot{\underline{x}} = B\underline{u} \ , \quad \underline{u} \in U \ ,$$

$$\dot{x}_i \leq 0 \ , \quad x_i \in I_p \ ,$$

and

$$\dot{x}_i = 0 \ , \quad x_i \in B_p$$

is a subset of the collection of all optimal solutions of the same problem, with $\{\lambda_i = 1, \quad x_i \in I_p\}$.

□ Proposition 2.3

2.5.3 Geometrical Characterization of the Feedback Space

The feedback solution to the optimal control problem, as it appears in [G4], [G5], [G6] and [G2], is based on the construction of regions in the state space such that to each region corresponds an optimal control vector. The set of all these regions covers the state space. Therefore the corresponding optimal controls are the feedback solution.

In [G4] an algorithm is presented for construction of regions with the property that to all initial conditions lying in a given region correspond the same set of optimal controls and the same boundary sequence. The geometrical characterization of these regions is a result of the following theorem:

Theorem 2.3 (see [G4], pp. 114-115):

The feedback-control-regions are convex polyhedral cones in R^n.

□ Theorem 2.3

The algorithm constructs the feedback-control-regions backwards in time, where in each stage it allows a set of nodes to "leave the boundary" (that is, a set of nodes is allowed to be different from zero from a given time until $-\infty$, and the other nodes remain at zero level all the time). (See [G4], pp. 115-135.)

In single destination networks with unity weightings in the cost functional there are properties that facilitate the construction of the regions. Some of these properties are:

Property 2.1 (see [G4], pp. 126-132, pp. 259-260, and [G2], pp. 84-86)

For single destination networks with unity weightings in the cost functional, there are no break walls in the state space. That is, the solution of one stage of the algorithm for a given set of nodes that leave the boundary backwards in time is constant in the entire time interval $(t_p,-\infty)$. Where t_p is the time when the set of nodes leave the boundary.

□ Property 2.1

Property 2.2

We conclude immediately from Property 2.1 that if we look at an optimal trajectory forward in time, there are switches only when a set of nodes reach the boundary (i.e., a set of state variables become zero).

□ Property 2.2

Chapter 3

Representation of the Optimal Control Problem as a Nonconvex Quadratic Program

3.1 Introduction

In the previous chapter it was proved that there is always an optimal solution to Problem 2.1 with piecewise constant controls and piecewise constant slopes in the state trajectory; therefore, this trajectory may be represented by a finite number of parameters (see Conclusion 2.1).

In this chapter, Problem 2.1 will be formulated in such a way that the optimization will be done over this finite set of parameters with the optimal solution fulfilling Proposition 2.2. This formulation (Problem 3.1) is a cubic static optimization problem. Solving such problems turns out to be quite difficult, but in Section 3.3 we simplify this problem by reducing it to a quadratic program (Problem 3.2). By means of this representation of the problem we shall try, in the following chapters, to solve the open loop dynamic routing problem.

3.2 Representation of the Optimal Control Problem by Means of the Controls and the Time Between Switches

3.2.1 The Optimal Control Problem

In Chapter 2 it was proved that there is always an optimal solution to Problem 2.1 with piecewise constant controls and with piecewise constant slopes in the state trajectory (see Conclusion 2.1).

Based on this fact we may formulate Problem 2.1 as a static optimization problem, with the controls $\underline{u}_i$, $i \in F$, and the time periods τ_i, $i \in F$ as the optimization variables. Solution of this optimization problem will provide an optimal solution satisfying Conclusion 2.1.

Performance Index

From (2.4) we see that it is possible to represent the performance index J as the area under the curve $\underline{\alpha}^T\underline{x}(t)$ in the plane with axes $\underline{\alpha}^T\underline{x}$ and t (from now on, this plane will be denoted by Z).

We will also define:

Definition 3.1

$J_i \triangleq$ {The cost functional in the time interval $[t_{i-1}, t_i)$, $i \in F$} .

As mentioned above, the state trajectories have piecewise constant slopes; then, since $\underline{\alpha}$ is constant, $\underline{\alpha}^T\underline{x}(t)$ also has piecewise constant slopes (see Figure 3.1).

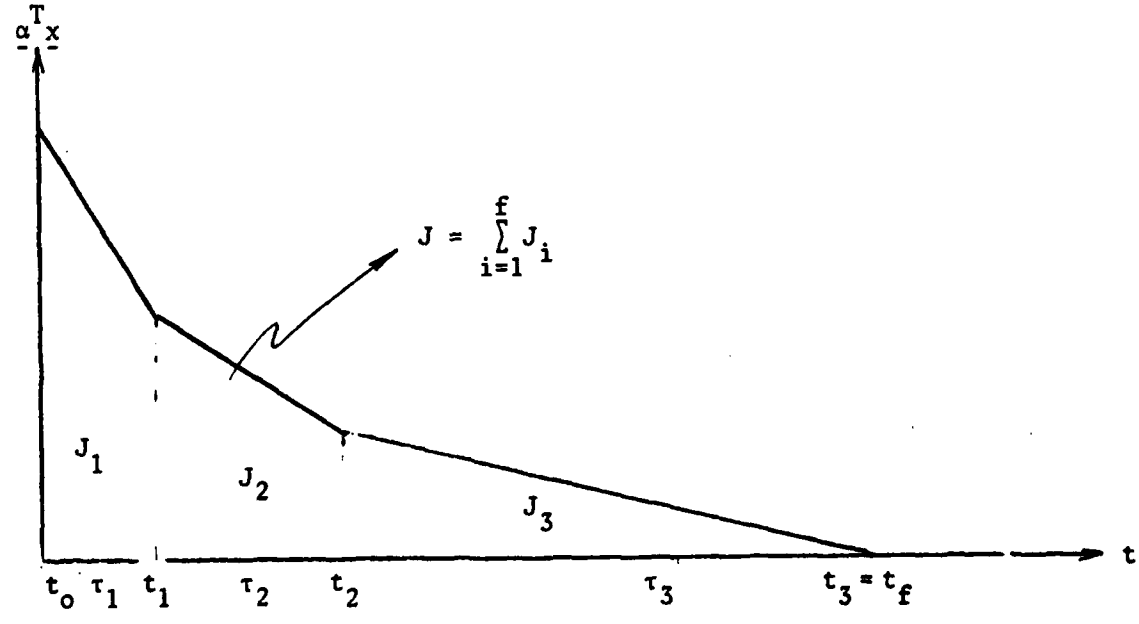


Figure 3.1: Trajectory with piecewise constant slope, in the Z-plane.

The slopes in each time interval are

$$\dot{\underline{x}}_p = \frac{\underline{x}(t_p) - \underline{x}(t_{p-1})}{\tau_p} \quad , \quad p \in F \quad , \tag{3.1}$$

then, from (2.5) and Proposition 2.1

$$\underline{x}(t_p) = \underline{x}(t_{p-1}) - B\underline{u}_p\tau_p \quad , \quad p \in F \quad . \tag{3.2}$$

We conclude from (3.2) that the knowledge of $\underline{x}_o$, τ_p and $\underline{u}_p$ for $p \in F$ is enough for knowing $\underline{x}(t)$, t. Likewise, (3.2) tells us that the values of the state variables at the switching times are:

$$\underline{x}(t_1) = \underline{x}(t_o) + B\underline{u}_1\tau_1 \ ,$$

$$\underline{x}(t_2) = \underline{x}(t_1) + B\underline{u}_2\tau_2 = \underline{x}(t_o) + B\underline{u}_1\tau_1 + B\underline{u}_2\tau_2 \ ,$$

$$\vdots \qquad \vdots \qquad \vdots$$

$$\underline{x}(t_f) = \ldots\ldots\ldots\ldots = \underline{x}(t_o) + B\underline{u}_1\tau_1 + \ldots\ldots + B\underline{u}_f\tau_f \ ,$$

that is to say

$$\underline{x}(t_p) = \underline{x}_o + \sum_{i=1}^{p} B\underline{u}_i\tau_i \quad , \quad p \in F \quad . \tag{3.3}$$

The performance index J will be calculated by calculating J_p for each $p \in F$ (see Figure 3.1). Note that J_p is equal to the area of the trapezoid in Figure 3.2, therefore

$$J_p = \int_{t_{p-1}}^{t_p} \underline{\alpha}^T\underline{x}(t)dt = \frac{1}{2}\underline{\alpha}^T[\underline{x}(t_p) + \underline{x}(t_{p-1})]\tau_p \quad , \quad p \in F. \tag{3.4}$$

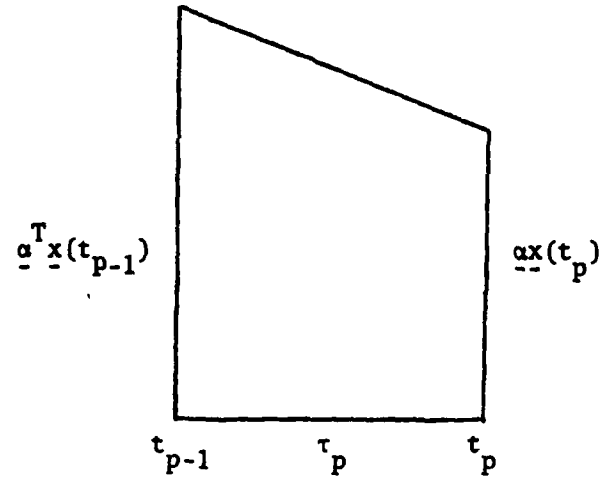


Figure 3.2: J_p as the area of a trapezoid.

From (3.3) and (3.4) we conclude that:

$$J_p = \frac{1}{2}\underline{\alpha}^T[\underline{x}_o + \sum_{i=1}^{p} \underline{B}\underline{u}_i\tau_i + \underline{x}_o + \sum_{i=1}^{p-1} \underline{B}\underline{u}_i\tau_i]\tau_p \quad , \quad p \in F \quad , \qquad (3.5)$$

or, after algebraic manipulations

$$J_p = \underline{\alpha}^T\underline{x}_o\tau_p + \underline{\alpha}^T\left\{\sum_{i=1}^{p-1} \underline{B}\underline{u}_i\tau_i\tau_p + \frac{1}{2}\underline{B}\underline{u}_p\tau_p^2\right\} \quad , \quad p \in F \quad , \qquad (3.6)$$

and the performance index J is (see Figure 3.1)

$$J = \sum_{p=1}^{f}\left\{\underline{\alpha}^T\underline{x}_o\tau_p + \underline{\alpha}^T\left[\sum_{i=1}^{p-1} \underline{B}\underline{u}_i\tau_i\tau_p + \frac{1}{2}\underline{B}\underline{u}_p\tau_p^2\right]\right\} \quad . \qquad (3.7)$$

End Condition

Based on (3.3), the end condition $\underline{x}(t_f) = 0$ can be written as:

$$\underline{x}_o + \sum_{i=1}^{f} \underline{B}\underline{u}_i\tau_i = 0 \quad . \qquad (3.8)$$

State and Control Constraints

The positivity of the state variables is assured by the following proposition:

Proposition 3.1

In the trajectories satisfying Conclusion 2.1, $\underline{x}(t) \geq 0$ for $t \in [t_o, t_f]$ iff $\underline{x}(t_p) \geq 0$, $p \in F$.

Proof

If $\underline{x}(t_p) \geq 0$ for $p \in F$ then $\underline{x}(t) \geq 0$ for $t \in [t_o, t_f]$ (since the slopes of the state trajectory are constant over the interval $[t_{p-1}, t_p)$, $p \in F$). On the other hand, it is obvious that if $\underline{x}(t) \geq 0$, $\forall t \leq t_f$, then $\underline{x}(t_p) \geq 0$, $p \in F$.

□ Proposition 3.1

From Proposition 3.1 and (3.3) follows that it is possible to write the constraint $\underline{x}(t) \geq 0$ as

$$\sum_{i=1}^{p} \underline{B}\underline{u}_i \tau_i \geq -\underline{x}_o \quad , \quad p = 1,2,\ldots,f-1 \quad . \tag{3.9}$$

The control constraints (2.7) remain without change for all the controls $\underline{u}_i$, $i \in F$.

In addition to these constraints, it is required that

$$\tau_i \geq 0 \quad , \quad i \in F \quad , \tag{3.10}$$

for assuring that $t_f \geq t_{f-1} \geq t_{f-2} \cdots \geq t_o$.

From (3.8) and (3.9) it is obvious that for each time interval $[t_{i-1}, t_i)$, $i \in F$, there is a change of $\underline{B}\underline{u}_i \tau_i$ in the state vector value; therefore in $[t_{i-1}, t_i)$ we have $\dot{\underline{x}} = \underline{B}\underline{u}_i$. Then the differential constraints (2.5) hold.

Based on the above, we can state the optimal dynamic routing problem as the following static optimization problem:

Problem 3.1

Find

$$\underline{u}_i \quad \text{and} \quad \tau_i \,, \quad i \in F \,, \tag{3.11}$$

that minimize the cost function

$$J = \sum_{p=1}^{f} \left\{ \underline{\alpha}^T \underline{x}_o \tau_p + \underline{\alpha}^T \left[\sum_{i=1}^{p-1} \underline{B}\underline{u}_i \tau_i \tau_p + \frac{1}{2} \underline{B}\underline{u}_p \tau_p^2 \right] \right\} \,, \tag{3.12}$$

subject to the constraints:

$$\underline{u}_i \geq 0, \quad \tau_i \geq 0, \quad i \in F \,, \tag{3.13}$$

$$\underline{D}\underline{u}_i \leq \underline{C} \,, \quad i \in F \,, \tag{3.14}$$

$$\sum_{i=1}^{p} \underline{B}\underline{u}_i \tau_i \geq -\underline{x}_o \,, \quad p = 1,2,\ldots,f-1 \,, \tag{3.15}$$

$$\sum_{i=1}^{f} \underline{B}\underline{u}_i \tau_i = -\underline{x}_o \,, \tag{3.16}$$

with the amount of switches $f-1$ free.

□ Problem 3.1

3.2.2 Properties of the Proposed Problem

As mentioned above, Problem 3.1 is a static formulation of Problem 2.1. In light of this fact, it is possible to solve the optimal dynamic routing problem by using known methods from static optimization.

There are general and efficient methods for special problems only, but there are no general methods for solving any nonlinear optimization problem; i.e., for each nonlinear optimization problem we must find the adequate method.

The cost function of Problem 3.1 is cubic (see (3.12)) and its constraint region is quadratic (see (3.15) and (3.16)), i.e., Problem 3.1 is a cubic optimization problem. From (3.16) we see immediately that the constraint region is nonconvex since Definition A.1 does not hold (see Example 3.1).

A problem with these properties is very hard to solve. In the next section we will transform Problem 3.1 to a quadratic one by changing the variables $\underline{u}_i$ and τ_i to new variables.

Example 3.1

Refer to the network of Figure 3.3. It is obvious that there are no switches in the optimal solution, then constraint (3.16) can be written as:

$$u\tau = -x_o \quad .$$

The constraint region is convex if for $u_1\tau_1 = -x_o$ and $u_2\tau_2 = -x_o$ then $[\lambda u_1 + (1-\lambda)u_2][\lambda\tau_1 + (1-\lambda)\tau_2] = -x_o$, $\forall\lambda \in [0,1]$. But $[\lambda u_1 + (1-\lambda)u_2][\lambda\tau_1 + (1-\lambda)\tau_2] = -\underline{x}_o + [-2\lambda u_2\tau_2 + \lambda u_1\tau_2 - \lambda^2 u_1\tau_2 + \lambda u_2\tau_1 - \lambda^2 u_2\tau_1 - 2\lambda^2 x_o]$ and obviously this expression is not equal to $-x_o$ for all $\lambda \in [0.1]$.

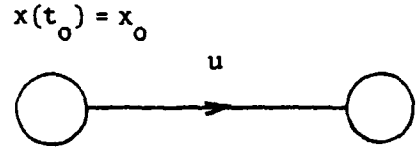


Figure 3.3: Network for Example 3.1

□ Example 3.1

3.3 The Optimal Control Problem as a Quadratic Program

3.3.1 The Optimal Control Problem

It can be easily noted that in all the quadratic and cubic elements of Problem 3.1 there is a term of the type $\underline{u}_i\tau_i$. Therefore it is possible

to transform Problem 3.1 into a problem with a lower degree by defining the following new variables:

Definition 3.2

$$\underline{z}_i \triangleq \underline{u}_i \tau_i \ , \quad i \in F \ .$$

(In the next section there is an explanation about the meaning of $\underline{z}_i$.)

Based on the definition above the cost function (3.12) can be written as:

$$J = \sum_{p=1}^{f} \left\{ \underline{\alpha}^T \underline{x}_o \tau_p + \underline{\alpha}^T \left[\sum_{i=1}^{p-1} B\underline{z}_i \tau_p + \frac{1}{2} B\underline{z}_p \tau_p \right] \right\} \ . \tag{3.17}$$

Likewise, the constraints (3.15) and (3.16) can be written as

$$\sum_{i=1}^{p} B\underline{z}_i \geq -\underline{x}_o \ , \quad p = 1,2,\ldots,f-1 \ , \tag{3.18}$$

and

$$\sum_{i=1}^{f} B\underline{z}_i = -\underline{x}_o \ , \tag{3.19}$$

respectively.

The constraints (3.13) become

$$\underline{z}_i \geq 0, \quad \tau_i \geq 0, \quad i \in F \ , \tag{3.20}$$

since, if (3.20) holds, from Definition 3.2 $\underline{u}_i \geq 0$. On the other hand if (3.13) holds, then $\underline{z}_i \geq 0$.

If we replace $\underline{u}_i$ from Definition 3.2 in (3.14), the capacity constraints become

$$D\underline{z}_i - C\tau_i \leq 0 \ , \quad i \in F \ . \tag{3.21}$$

In light of these facts, the optimal dynamic routing problem may be formulated as:

Problem 3.2

Find

$$\underline{z}_i \quad \text{and} \quad \tau_i \,, \qquad i \in F \tag{3.22}$$

(with the amount of switches $f-1$ not known) that minimize

$$J = \sum_{p=1}^{f} \left\{ \underline{\alpha}^T \underline{x}_o \tau_p + \underline{\alpha}^T \left[\sum_{i=1}^{p-1} B\underline{z}_i \tau_p + \frac{1}{2} B\underline{z}_p \tau_p \right] \right\} \,, \tag{3.23}$$

subject to the constraints

$$z_i \geq 0, \quad \tau_i \geq 0, \quad i \in F \,, \tag{3.24}$$

$$D\underline{z}_i - \underline{C}\tau_i \leq 0 \,, \quad i \in F \,, \tag{3.25}$$

$$\sum_{i=1}^{p} B\underline{z}_i \geq -\underline{x}_o \,, \quad p = 1,2,\ldots,f-1 \,, \tag{3.26}$$

$$\sum_{i=1}^{f} B\underline{z}_i = -\underline{x}_o \,, \tag{3.27}$$

□ Problem 3.2

3.2.2 Meaning of the $\underline{z}_p$-variables

If we replace $\underline{u}_p \tau_p$ by $\underline{z}_p$, expression (3.2) becomes

$$\underline{x}(t_p) = \underline{x}(t_{p-1}) + B\underline{z}_p \tag{3.28}$$

i.e., $B\underline{z}_p$ is the value of the change in the state variable vector $\underline{x}$ during τ_p. If we look only at one element of the vector $\underline{z}_p$ (denoted by $(z_{ik}^j)^p$).

$$(z_{ik}^j)^p = (u_{ik}^j)^p \cdot \tau_p \,. \tag{3.29}$$

Then $(z_{ik}^j)^p$ is the value of the change in x_i and x_k (on opposite directions) because of the information flow from node i to node k destinated to node j, during τ_p (denoted by $(u_{ik}^j)^p$).

3.3.3 Properties of Problem 3.2

The optimal dynamic routing problem is formulated as a static optimization problem in Problem 3.2. Its cost function is quadratic (see (3.23)), and the constraint region is linear (see (3.24) - (3.27)); i.e., Problem 3.2 is a quadratic program.

Much is known about quadratic programming in comparison with other non-linear optimization problems, mainly when the cost function is convex, concave, quasi-convex, pseudo-convex or quasi-concave (the definition of these terms and an explanation of their importance appear in Appendix A).

As a result of that, we must determine whether or not one of these properties holds.

From Proposition A.1 it is clear that if a specific function is not quasi-convex, then it is not pseudo-convex or convex. Hence, we will show that these three properties do not hold by showing an example with a non-quasi-convex function. This example is based on the following network:

Network 3.1 (see [G4], pp. 93-104):

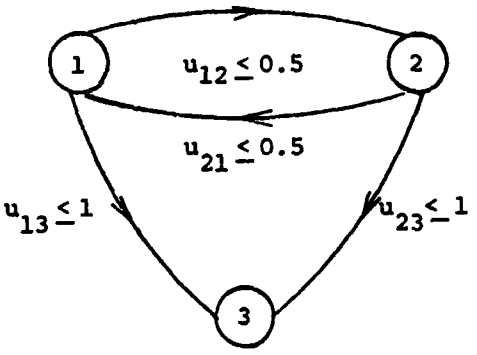


Figure 3.4: Network 3.1

In Network 3.1, as it appears in Figure 3.4, there is only one destination node (node 3). Then it is possible to omit the index that represents the destination node in the controls and the state variables. In light of these facts, the upper index will denote the time interval in

question; for example u_{12}^1 is the control u_{12} during the time interval $[t_o,t_1)$ (we will use this notation henceforth when referring to single destination networks).

From Figure 3.4 it is clear that the differential equations (2.5) for Network 3.1 are

$$\begin{aligned} \dot{x}_1(t) &= -u_{13}(t) - u_{12}(t) + u_{21}(t) \\ \dot{x}_2(t) &= -u_{23}(t) + u_{12}(t) - u_{21}(t) \end{aligned} \quad , \tag{3.30}$$

likewise, if the cost function weighting constants are all 1, (3.23) turns to

$$J = \sum_{p=1}^{f} \left\{ (x_{o1}+x_{o2})\tau_p + \sum_{i=1}^{p-1} (-z_{13}^i - z_{23}^i)\tau_p + \frac{1}{2}(-z_{13}^p - z_{23}^p)\tau_p \right\} , \tag{3.31}$$

with x_{oi} being the i-th element of the initial condition vector, and z_{ik}^p being z_{ik} during the time interval $[t_{p-1},t_p)$.

□ Network 3.1

Example 3.2

Referring to Network 3.1 with the initial condition $\underline{x}_o = (8,2)^T$, we will denote

$$\underline{z}_p = (z_{12}^p, z_{21}^p, z_{13}^p, z_{23}^p) \quad .$$

It is easy to verify that the solution (denoted by $\underline{w}^1$)

$$\left.\begin{aligned} \underline{z}_1 &= (2,0,3,3) \ , \quad \tau_1 = 9 \\ \underline{z}_2 &= (1,0,2,2) \ , \quad \tau_2 = 16 \end{aligned}\right\} \ \underline{w}_1$$

is feasible, and that $J(\underline{w}^1) = 95$.

On the other hand, the solution $\underline{w}^2$

$$\left.\begin{array}{ll} \underline{z}_1 = (0,0,2,1) , & \tau_1 = 8 \\ \underline{z}_2 = (2,0,4,3) , & \tau_2 = 8 \end{array}\right\} \underline{w}_2$$

is also feasible and for it the cost function is $J(\underline{w}^2) = 96$.

Referring to those two solutions

$$J(\lambda\underline{w}^1 + (1-\lambda)\underline{w}^2) = 98.875 \quad \text{for} \quad \lambda = 1/2 \ ,$$

that is to say

$$J(\lambda\underline{w}^1 + (1-\lambda)\underline{w}^2) \not\leq \max\{J(\underline{w}^1), J(\underline{w}^2)\} \ .$$

We then conclude that Definition A.5 does not hold for all $\lambda \in [0,1]$.

□ Example 3.2

In a similar way it is possible to show that in general the cost function is not quasi-concave (see Example 3.3).

Example 3.3

Referring to Network 3.1 with the initial condition $\underline{x}_o = (4,1)^T$, it is easy to verify that the solution (denoted by $\underline{w}_1$)

$$\left.\begin{array}{ll} \underline{z}_1 = (0,0,0.5,0.5) , & \tau_1 = 5 \\ \underline{z}_2 = (1.5,0,2,2) , & \tau_2 = 5 \end{array}\right\} \underline{w}^1$$

is feasible and for it the cost function is $J(\underline{w}^1) = 32.5$.

On the other hand, the solution $\underline{w}^2$

$$\left.\begin{array}{ll} \underline{z}_1 = (0.4,0,0.6,0.5) , & \tau_1 = 5.595 \\ \underline{z}_2 = (0,0,3,0.9) , & \tau_2 = 3.905 \end{array}\right\} \underline{w}^2$$

is also feasible and for it the cost function is $J(\underline{w}^2) = 32.5125$.

Based on these two solutions

$$J(\lambda \underline{w}^1 + (1-\lambda)\underline{w}^2) = 32.49 \quad \text{for} \quad \lambda = 3/4 \quad ,$$

that is to say

$$J(\lambda \underline{w}^1 + (1-\lambda)\underline{w}^2) \not\geq \min\{J(\underline{w}^1), J(\underline{w}^2)\} \quad .$$

We then conclude that Definition A.8 does not hold for all $\lambda \in [0,1]$.

□ Example 3.3

It is possible to resume this section with the following proposition:

Proposition 3.2

Problem 3.2 is a nonconvex quadratic program, with a cost function that is not concave, quasi-convex, quasi-concave or pseudo-convex even for the case of single destination networks with unity weightings in the cost function.

□ Proposition 3.2

In Appendix B there is a general review of the methods for solving nonconvex quadratic programs, and in Chapter 5 we give propositions for solving Problem 3.2 by using these methods.

In the next chapter we will solve Problem 3.2 for single destination networks with unity weightings in the cost function by using special properties of these networks.

Chapter 4

Open-Loop Solutions for the Dynamic Routing Problem for Single Destination Networks With Unity Weightings in the Cost Functional by Using Linear Programming

4.1 Introduction

In this chapter we will develop a theory that will enable us to propose an algorithm for solving the open-loop optimal dynamic routing problem for single destination networks with unity weightings in the cost functional.

In Section 4.2 the theoretical background is developed, and in Section 4.3 the algorithm is presented (Section 4.3.1) and illustrated by examples (Section 4.3.2).

Single destination networks with unity weightings in the cost functional have properties that permit the solution of the optimal dynamic open-loop routing problem by using linear programming.

The necessary theory for developing the algorithm will be based on properties from Chapter 2, also the cost function representation in the Z-plane will be used (see Figure 3.1). Since it is known that there always exists an optimal solution with piecewise constant controls, we shall search only for an optimal solution with this property.

Unity weightings in the cost function imply $\underline{\alpha} = \underline{1}$, therefore

$$\underline{\alpha}^T \underline{x} = \underline{1}^T \underline{x} = \sum_{i=1}^{n} \dot{x}_i$$
$$\underline{\alpha}^T \dot{\underline{x}} = \sum_{i=1}^{n} \dot{x}_i \qquad (4.1)$$

From the linear program (2.20) and from Proposition 2.3 follows that for single destination networks with unity weightings in the cost functional, the optimal solution is a solution of the following linear program:

$$\min \sum_{i=1}^{n} \dot{x}_i(t)$$

subject to

$$\dot{x}_i \leq 0 \ , \quad \forall x_i \in I_p$$

and

$$\dot{x}_i = 0 \ , \quad \forall x_i \in B_p \ ,$$

then, since $\sum_{i=1}^{n} \dot{x}_i(t)$ is the slope of the solution in the Z-plane (see Figure 4.1):

Proposition 4.1

The slope in the Z-plane of the optimal solution is minimal among all possible slopes satisfying

$$\dot{x}_i \leq 0 \ , \quad \forall x_i \in I_p$$

$$\dot{x}_i = 0 \ , \quad \forall x_i \in B_p \ .$$

Observe that since the slope is always negative, "minimal slope" means "the steepest".

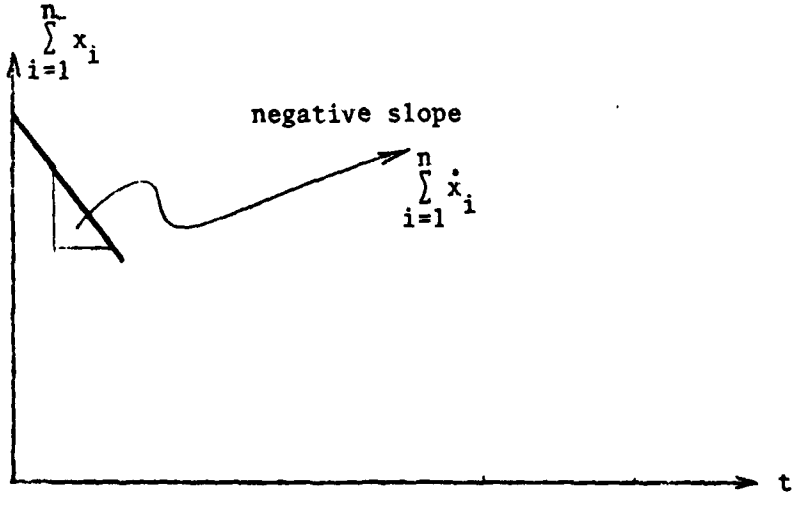


Figure 4.1: The slope in the Z-plane.

□ Proposition 4.1

Generally we will represent trajectories $(x(t)\ ,\ t \in [t_o, t_f])$ in the Z-plane, since this allows us to express the optimal cost as the area under the curve representing the trajectory.

Note that, for a given time, a point in the Z-plane represents all the state vectors with the same value of $\sum_{i=1}^{n} x_i$.

We conclude from Proposition 4.1 that:

Conclusion 4.1

In single destination networks with unity weightings in the cost functional there is no feasible trajectory with initial slope in the Z-plane smaller than the slope of the optimal solution (see Figure 4.2).

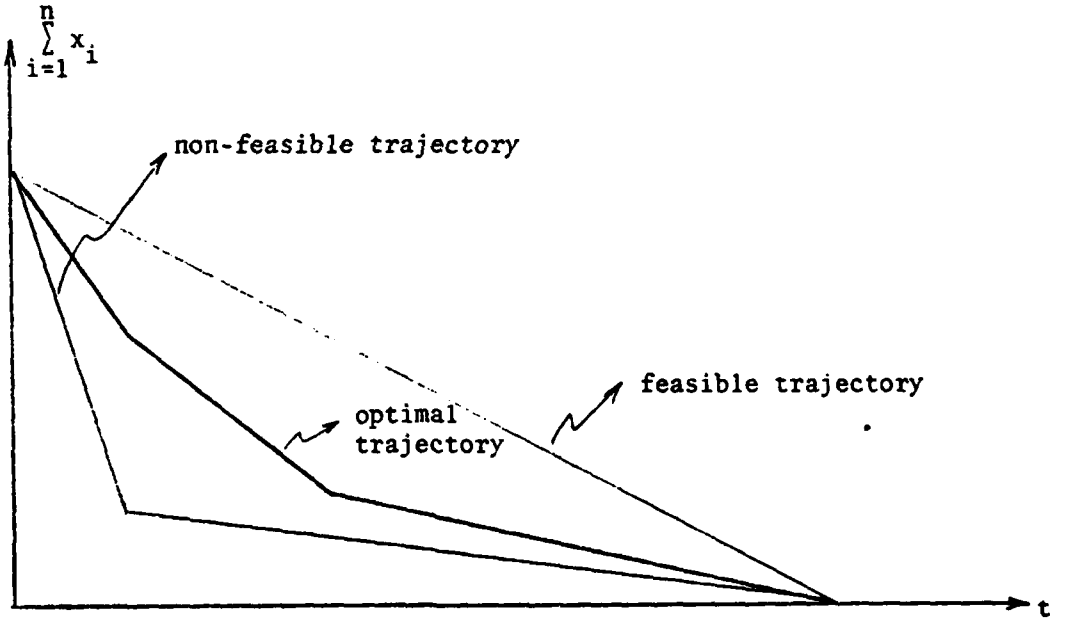


Figure 4.2: Initial slope of the optimal trajectory in the Z-plane.

□ Conclusion 4.1

4.2 Theoretical Background

Referring to a feasible solution with ℓ switches ($\ell \leq f - 1$ where $f - 1$ is the number of switches in the optimal solution), we define:

Definition 4.1

The feasible solution with minimal cost function among all the feasible solutions with ℓ (or less) switches will be called the "ℓ-switches-optimal-solution".

□ Definition 4.1

The algorithm to be developed in Section 4.3 is based on the following lemmas and theorems:

Lemma 4.1

If there exists a one-switch trajectory passing from $\underline{x}_o$ to $\underline{x}_f$ in a time period t_f, then there exists a non-switch trajectory connecting these two state vectors in the same time period (see Figure 4.3).

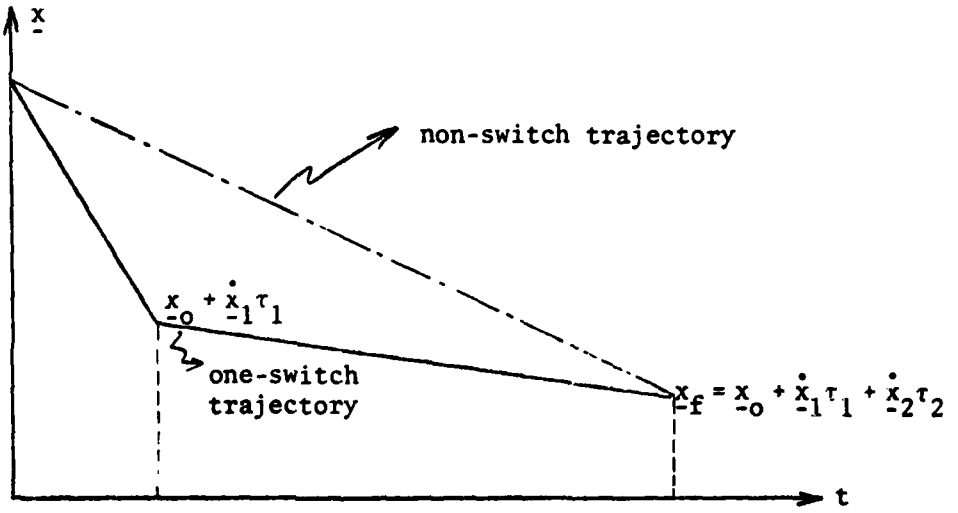


Figure 4.3: State trajectories for Lemma 4.1

Proof

We will denote the controls in the one-switch trajectory as follows: u_i^1, $i = 1,\ldots,m$ for $t \in [t_o,t_1)$, and u_i^2, $i = 1,\ldots,m$ for $t \in [t_1,t_f]$ (see Figure 4.3).

The trajectory received from the application of these controls is feasible, therefore

$$u_i^1 \leq C_i \quad \text{and} \quad u_i^2 \leq C_i \ , \quad i = 1,\ldots,m \ . \tag{4.2}$$

We now refer to the trajectory received from the application of the controls

$$u_i = \frac{u_i^1\tau_1 + u_i^2\tau_2}{\tau_1 + \tau_2} \ , \quad i = 1,\ldots,m \ , \tag{4.3}$$

during the time period t_f, $(t_f = \tau_1+\tau_2)$ starting from the initial state vector $\underline{x}_o$.

Feasibility Test

(a) $u_i \geq 0$, $\forall i = 1,\ldots,m$ since (4.3) implies that u_i is a sum of positive elements.

(b) From (4.3) it is obvious that

$$u_i \leq \frac{\max\{u_i^1,u_i^2\}\tau_1 + \max\{u_i^1,u_i^2\}\tau_2}{\tau_1 + \tau_2} \ ,$$

then

$$u_i \leq \max\{u_i^1,u_1^2\} \ ,$$

therefore, from (4.2)

$$u_i \leq C_i \ , \quad i = 1,\ldots,m \ .$$

Value of the State Vector at t_f

Since $\dot{\underline{x}}$ is a linear combination of the elements of the control vector (see (2.29)), it follows from (4.3) that

$$\dot{\underline{x}} = \frac{\dot{\underline{x}}_1\tau_1 + \dot{\underline{x}}_2\tau_2}{\tau_1 + \tau_2} \ , \tag{4.4}$$

where $\dot{\underline{x}}_1$ and $\dot{\underline{x}}_2$ are the state rates of the one-switch trajectory before

and after the switch respectively.

At t_f the state is $\underline{x}_o + \dot{\underline{x}} t_f$; then, from (4.4) follows that the state is $\underline{x}_o + \dot{\underline{x}}_1 \tau_1 + \dot{\underline{x}}_2 \tau_2$, i.e. $\underline{x}_f$ (see Figure 4.3).

In light of these facts, we see that the controls defined in (4.3) determine a non-switch trajectory connecting $\underline{x}_o$ and $\underline{x}_f$ in a time period t_f.

□ Lemma 4.1

Based on Lemma 4.1, the following theorem can be proved:

Theorem 4.1

If there exists a trajectory with $f-1$ switches, taking $\underline{x}_o$ to $\underline{x}_f$ in time t_f $(t_f = \sum_{i=1}^{f} \tau_i)$, then there exists a non-switch trajectory connecting these two state vectors in the same time period.

Proof

We will prove the theorem by repeated use of Lemma 4.1.

Suppose that there are $f-1$ switches in the trajectory. Then denote by $\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_{f-1}$ the state vectors at $t_1, \ldots, t_{f-1}$ respectively.

Based on Lemma 4.1, we construct a non-switch trajectory passing from $\underline{x}_o$ to $\underline{x}_2$ in a time period $\tau_1 + \tau_2$. With this trajectory, by using again Lemma 4.1, we construct a non-switch trajectory passing from $\underline{x}_o$ to $\underline{x}_3$ in a time period $\tau_1 + \tau_2 + \tau_3$. By executing this procedure f times, we get a non-switch trajectory passing from $\underline{x}_o$ to $\underline{x}_f$ in a time period t_f (see Figure 4.4).

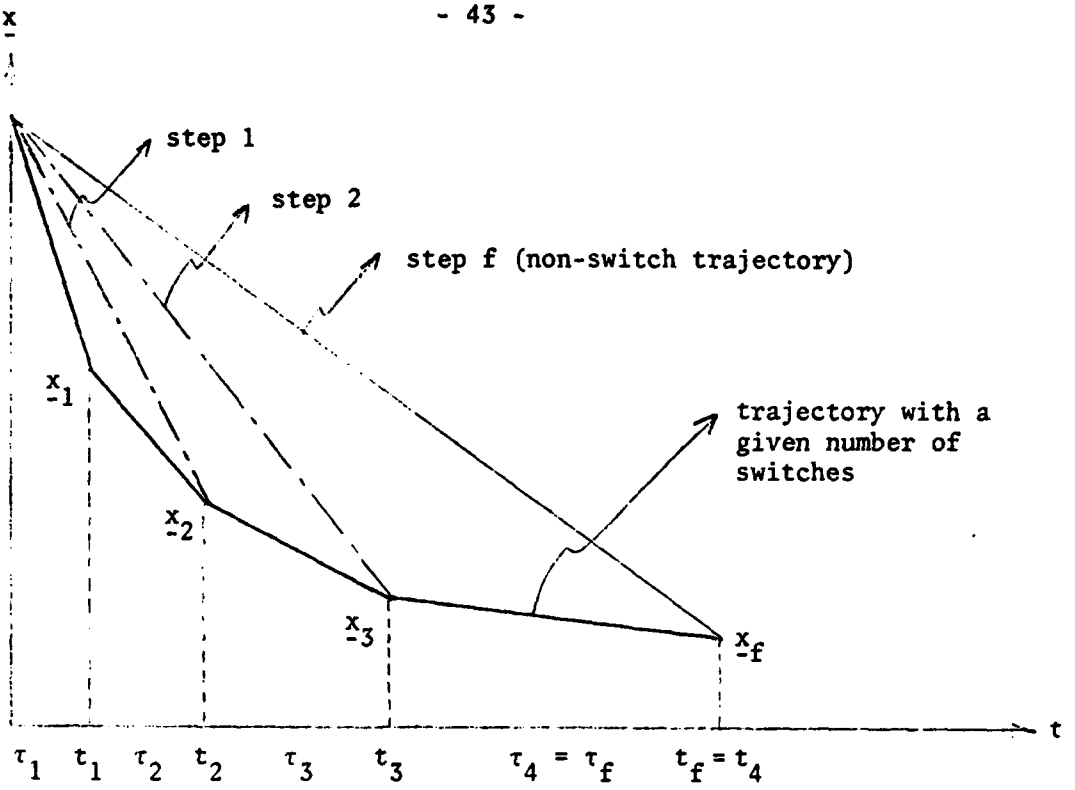


Figure 4.4: State trajectories for Theorem 4.1

□ Theorem 4.1

We conclude from Theorem 4.1 that:

Conclusion 4.2

If for a given initial condition $\underline{x}_o$ there exists a feasible solution that empties the network ($\underline{x}_f = \underline{0}$) in a time period t_f, then there exists a feasible solution without switches that empties the network in the same period.

□ Conclusion 4.2

All remaining theorems in this chapter refer to single destination networks with unity weightings in the cost functional.

We will prove first the following fundamental theorem:

Theorem 4.2

Any feasible trajectory $\hat{\underline{x}}(t)$ must satisfy

$$\sum_{i=1}^{n} \hat{x}_i(t) \geq \sum_{i=1}^{n} x_i^*(t) \quad ,$$

for all $t \in [t_o, t_f^*]$.

Proof

We can put in words the statement of the theorem as follows: there are no feasible trajectories in the Z-plane penetrating into the region underneath the optimal trajectory. Or by using Theorem 4.1: there are no feasible trajectories without switches in the Z-plane penetrating into the region underneath the optimal trajectory.

We will use the following notation (see Figure 4.5):

(a) $X(t) = \sum_{i=1}^{n} x_i(t)$

(b) $X_o = \sum_{i=1}^{n} x_i(t_o)$

(c) $x_d \triangleq$ {destination node}

We will prove the theorem by proving that there is no trajectory without switches, $\hat{\underline{x}}(t)$, such that $\hat{X}(t') < X^*(t')$ for some $t' \in [t_o, t_f^*]$.

We shall first prove the theorem for the case when the optimal solution has at most one switch.

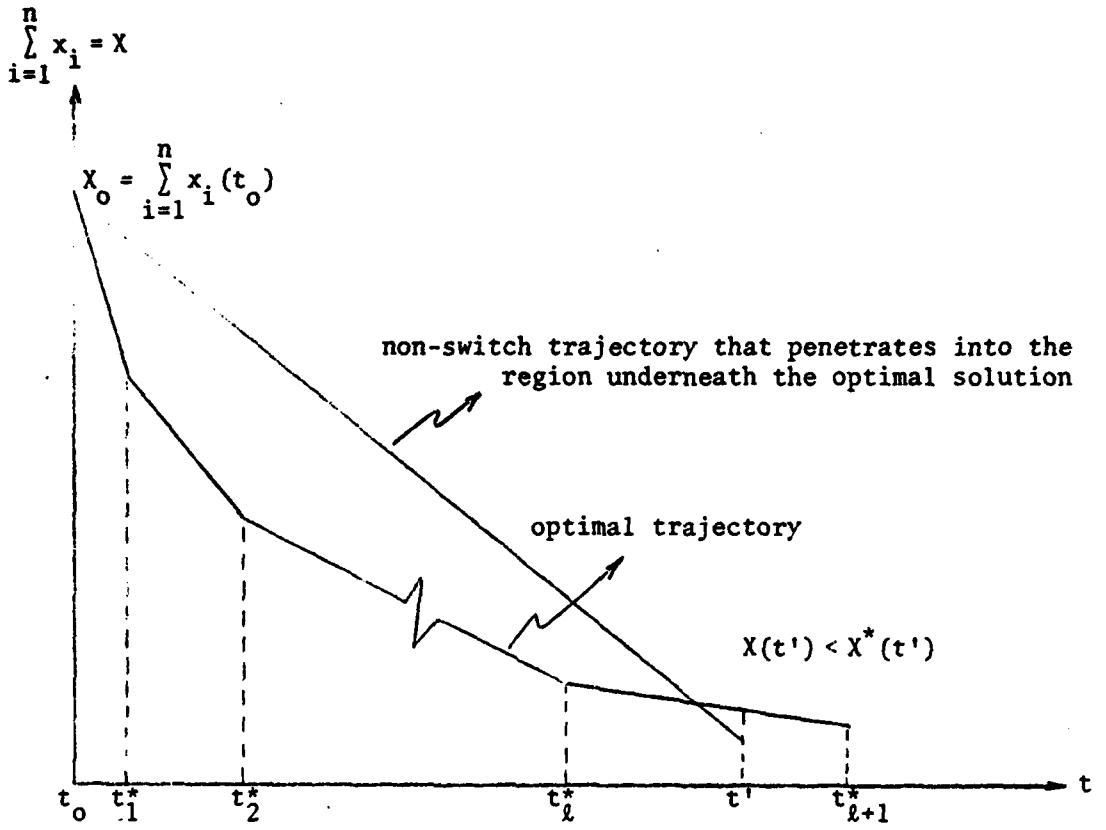


Figure 4.5: Trajectory without switches contradicting Theorem 4.2

Lemma 4.2

For optimal solutions with at most one switch, Theorem 4.2 holds.

Proof

From Conclusion 4.1 follows that if there are no switches in the optimal solution, then $\hat{X}(t) \geq X^*(t)$, $t \in [t_o, t_f^*]$, since the initial slope in the Z-plane of the optimal solution is the smallest feasible slope.

Suppose now that there is one switch in the optimal solution; from the above follows that $\hat{X}(t) \geq X^*(t)$ for $t \in [t_o, t_1^*)$. That is, if there exists a time t' satisfying $\hat{X}(t') < X^*(t')$, then $t' \in [t_1^*, t_f^*]$ (see Figure 4.6). Hence, we will prove the lemma by proving that $\hat{X}(t') \geq X^*(t')$, $\forall t' \in [t_1^*, t_f^*]$.

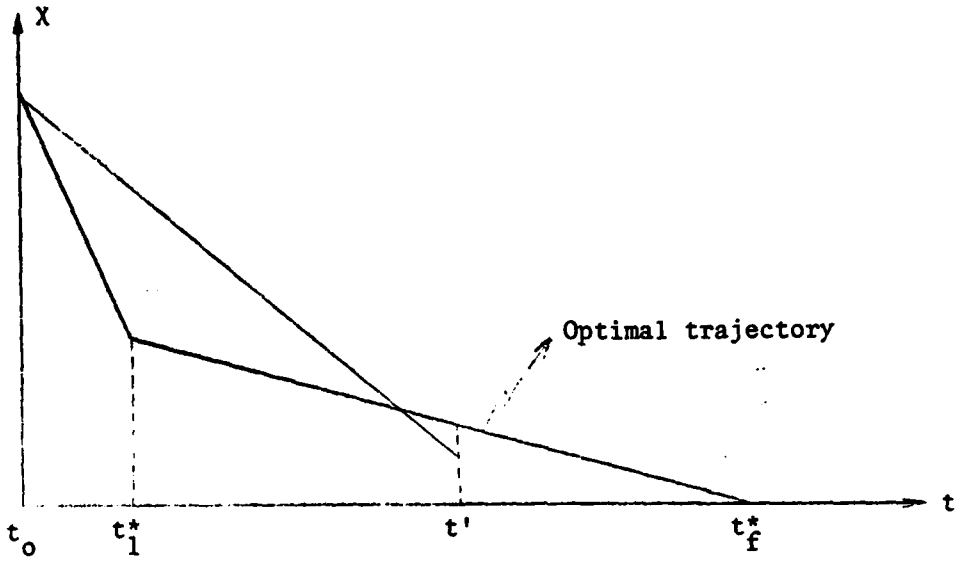


Figure 4.6: Trajectories in the Z-plane for Lemma 4.2

We shall refer only to optimal solutions that change the slope in the Z-plane at t_1^*, since for optimal solutions that do not change the slope at t_1^* a new optimal solution having no switches can be constructed (by direct application of Theorem 4.1). Note that a solution that has a switch and maintains a constant slope in the Z-plane is a solution that changes the controls but maintains $\sum_{i=1}^{n} \dot{x}_i$ without change.

From the above follows that there is a set of nodes i for which $x_i(t)$ vanish at t_1^* and remains zero afterwards, since otherwise it would be possible to continue after t_1^* with the same slope in the Z-plane, contradicting the assumption that there is a switch at t_1^*.

We denote:

$$M \triangleq \{x_i / x_i^*(t) = 0, \ \forall t \in [t_1^*, t_f^*], \ i = 1,\ldots,n\} ,$$

that is, M includes all the state variables in the optimal solution that are zero from t_o or from t_1^*.

$$K \triangleq \{x_i / x_i^* \notin M, \ i = 1,\ldots,n\} ,$$

that is, K includes all the state variables in the optimal solution that vanish not earlier than t_f^*.

Using this partition of the network we define:

Definition 4.2 (see Figure 4.7)

(a) $U_{Md}(t) \triangleq \sum_{\substack{j\in M \\ (j,d)\in L}} u_{jd}(t)$;

(b) $U_{Kd}(t) \triangleq \sum_{\substack{k\in K \\ (k,d)\in L}} u_{kd}(t)$,

(c) $U_{KM}(t) \triangleq \sum_{\substack{k\in K \\ (k,j)\in L}} \sum_{j\in M} u_{kj}(t) - \sum_{\substack{k\in K \\ (j,k)\in L}} \sum_{j\in M} u_{jk}(t)$.

Observe that U_{KM} is the net flow from K to M, i.e. the flow from K to M minus the flow from M to K.

(d) $C_{Kd} \triangleq \sum_{\substack{k\in K \\ (k,d)\in L}} C_{kd}$.

C_{Md} and C_{KM} are defined in a similar way:

(e) $X_K(t) \triangleq \sum_{k\in K} x_k(t)$,

$X_M(t) \triangleq \sum_{i\in M} x_i(t)$.

Likewise, we define $\dot{X}(t)$, $\dot{X}_K(t)$ and $\dot{X}_M(t)$ as the time derivatives of $X(t)$, $X_K(t)$ and $X_M(t)$ respectively.

(f) $X_{oK} \triangleq \sum_{k\in K} x_k(t_o)$,

$X_{oM} \triangleq \sum_{i\in M} x_i(t_o)$.

□ Definition 4.2

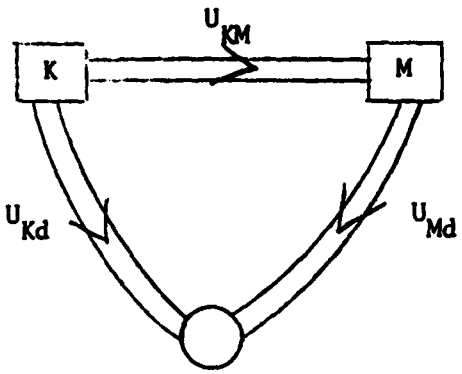


Figure 4.7: Partition of the network into the sets K and M

We will prove that $\hat{X}(t') \geq X^*(t')$, $\forall t' \in [t_1^*, t_f^*]$ by developing appropriate expressions for $\hat{X}(t')$ and $X^*(t')$. But first we show that $\dot{X}_K^*$ is minimal and constant on $[t_o, t_f^*]$. From Figure 4.7 it is obvious that

$$\dot{X}_K^* = -U_{Kd}^* - U_{KM}^* \quad , \tag{4.5}$$

hence, in order to show that $\dot{X}_K^*$ is minimal and constant we will show that U_{Kd}^* and U_{KM}^* are maximal and constant.

The nodes that belong to K vanish only at t_f^*, therefore the flow of information from these nodes to the destination node is always maximal, then

$$U_{Kd}^*(t) = C_{Kd} \quad , \quad \forall t \in [t_o, t_f^*] \quad . \tag{4.6}$$

For each node that belongs to M the net flow of information from K to M from t_o until the switch time, is smaller than or equal to the flow of information from the node to the destination node ($\dot{x} \leq 0$). Therefore, from $\dot{X}_M < 0$ follows that

$$U_{KM}^*(t) < U_{Md}^*(t) \quad , \quad t \in [t_o, t_1^*) \tag{4.7}$$

Likewise, it is obvious that $U_{KM}^*(t)$ for $t \in [t_o, t_1^*)$ is maximal, since otherwise it would be possible to increase it and in this way to obtain a trajectory such that $X(t) = X^*(t)$ for $t \in [t_o, t_1^*)$ and $X_M(t_1^*) > 0$, i.e.,

it would be possible to obtain a new trajectory having a slope smaller than the slope of the optimal solution after t_1^*, contradicting Proposition 4.1.

After the switch, X_M is equal to zero, therefore maximal rate of evacuation of the information from the network means maximal $\dot{X}_K$. Hence $U_{KM}^*(t)$ continues to be maximal after the switch. Note that the same maximal flow for $t \in [t_o, t_1^*)$ is possible since - as explained above - this flow was feasible when $\dot{x}_i \leq 0$ for $i \in M$, then it is obviously feasible when it is needed that $\dot{x}_i = 0$ for $i \in M$.

From the above follows that $U_{KM}^*(t)$ is constant and maximal on $[t_o, t_f^*]$. Therefore, it follows from (4.5) and (4.6) that $\dot{X}_K^*$ is constant and minimal on $[t_o, t_f^*]$.

We will use now this fact for proving the lemma.

From the definition of X follows that

$$X = X_K + X_M \quad , \tag{4.8}$$

likewise, it is known that if $\dot{\underline{x}}$ is constant on $[t_o, t']$

$$\underline{x}(t') = \underline{x}_o + \dot{\underline{x}} t' \quad , \tag{4.9}$$

where without loss of generality it is assumed that $t_o = 0$.

Using the facts that $\dot{X}_M^* = 0$ on $[t_1^*, t_f^*]$ and $\dot{X}_K^*$ is constant on $[0, t_f^*]$, it follows from (4.8) and (4.9) that

$$X^*(t') = X_{oK} + \dot{X}_K^* t' + X_{oM} + \dot{X}_M^* t_1^* \quad , \tag{4.10}$$

where

$$X_o = X_{oM} + X_{oK} \quad .$$

But the contents of nodes that belong to M vanish at t_1^*, then

$$X_M^*(t') = X_{oM} + \dot{X}_M^* t_1^* = 0 \quad . \tag{4.11}$$

By using (4.11), equation (4.10) may be written as

$$X^*(t') = X_{oK} + \dot{X}^*_K t' \ , \quad t' \in [t_1^*, t_f^*] \ . \tag{4.12}$$

The trajectory we are going to test has no switches, therefore

$$\hat{X}(t') = X_{oK} + \dot{\hat{X}}_K t' + X_{oM} + \dot{\hat{X}}_M t' \ , \tag{4.13}$$

but

$$\dot{\hat{X}}_M(t') = X_{oM} + \dot{\hat{X}}_M t' \ , \tag{4.14}$$

then, (4.13) may be written as

$$\hat{X}(t') = X_{oK} + \dot{\hat{X}}_K t' + \hat{X}_M(t') \ , \quad t' \in [t_1^*, t_f^*]. \tag{4.15}$$

It is known that $\hat{X}_M(t') \geq 0$ and, since $\dot{X}^*_K$ is minimal, $\dot{X}^*_K \leq \dot{\hat{X}}_K$. Therefore comparing (4.15) with (4.12) we conclude that

$$\hat{X}(t') \geq X^*(t') \ , \quad t' \in [t_1^*, t_f^*] \ .$$

□ Lemma 4.2

We will use now Lemma 4.2 in order to prove by induction the theorem for the general case. For this purpose we will suppose that Theorem 4.2 holds for an optimal solution having $h-1$ switches, and then we prove that the theorem also holds for an optimal solution having h switches.

For an optimal solution with h switches, there cannot be penetration in the Z-plane into the zone underneath the optimal trajectory for any time smaller than t_h^* (because the theorem holds for an optimal solution having $h-1$ switches), i.e., $\hat{X}(t) \geq X^*(t)$ for $t \in [t_o, t_h^*)$ where t_h^* is the h-switch in the optimal solution. It follows then that $\hat{X}(t)$ can be smaller than $X^*(t)$ only on $[t_h^*, t_f^*]$ (see Figure 4.8).

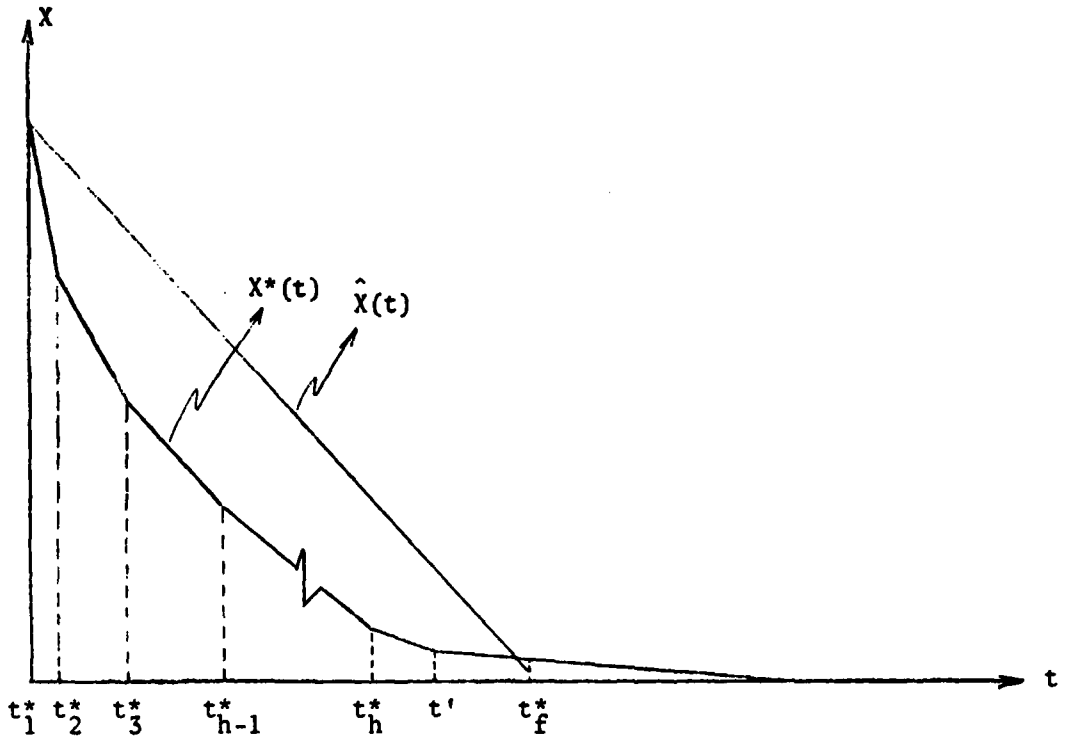


Figure 4.8: Trajectories in the Z-plane for proving Theorem 4.2 in the general case.

The proof will follow the same pattern as the proof of the lemma. Therefore we define the sets K and M, then we show that $\dot{X}^*_K$ is minimal and constant, and as a consequence of this, that $\hat{X}(t')$ is greater than or equal to $X^*(t')$.

We then define:

$K \triangleq$ {set of nodes whose contents vanish in the optimal solution exactly at t^*_f} ,

$M \triangleq$ {set of nodes whose contents vanish in the optimal solution before t^*_f} ,

that is, the set M includes all nodes that are empty from t_o, and those that are empty from $t^*_1, t^*_2, \ldots, t^*_h$.

In order to prove that $\dot{X}^*_K$ is minimal and constant, suppose that $\dot{X}^*_K$ is not minimal in some time interval. It is then possible to increase

the net flow from K to M in this interval. As a consequence of that, a trajectory that coincides with the optimal trajectory in the Z-plane until t_h^* such that $X_M(t_h^*) > 0$ is obtained. With this new trajectory it is possible to continue after t_h^* with a slope smaller than that of the optimal trajectory, contradicting Proposition 4.1. $\dot{X}_K^*$ is constant in $[t_o, t_f^*]$ because of the fact that the same maximal flow from K to M remains before and after every switch (note that at each switch some of the state variables that belong to M become zero). This is explained by the fact that if this maximal flow was possible before these states become zero (i.e., when in those nodes the derivative of the state was negative), it is obvious that it is still possible after the switch when these nodes are empty (i.e., when the derivative of the state is equal to zero).

By making the same steps as in Lemma 4.2, the same expressions for $X^*(t')$ and $\hat{X}(t')$ are obtained, that is:

$$X^*(t') = X_{oK} + \dot{X}_K^* t'$$

and

$$\hat{X}(t') = X_{oK} + \dot{\hat{X}}_K t' + \hat{X}_M(t') \ .$$

Since $\hat{X}_M(t')$ is nonnegative and $\dot{\hat{X}}_K \geq \dot{X}_K^*$, the theorem is proved by comparing these two expressions.

□ Theorem 4.2

Based on Theorem 4.2 the following theorems can be proved:

Theorem 4.3

Referring to solutions with piecewise constant controls. Each optimal solution to the minimum delay problem is also optimal to the minimum total time problem (i.e., min t_f).

Proof

From Theorem 4.1 follows that if there exists a trajectory with total time smaller than t_f^*, then there exists also a non-switch trajectory with total time τ smaller than t_f^*, i.e.,

$$\tau < t_f^* \quad . \tag{4.16}$$

Proposition 4.1 implies that the trajectory satisfying (4.16) has an initial slope greater than the slope of the optimal solution. Therefore, the fulfillment of (4.16) requires that $\sum_{i=1}^{n} x_i(t') < \sum_{i=1}^{n} x_i^*(t')$ for some t' (where $\underline{x}(t)$ is the trajectory satisfying (4.16)), contradicting Theorem 4.2 (see Figure 4.9). Hence, t_f^* is the optimum of the minimum total time problem.

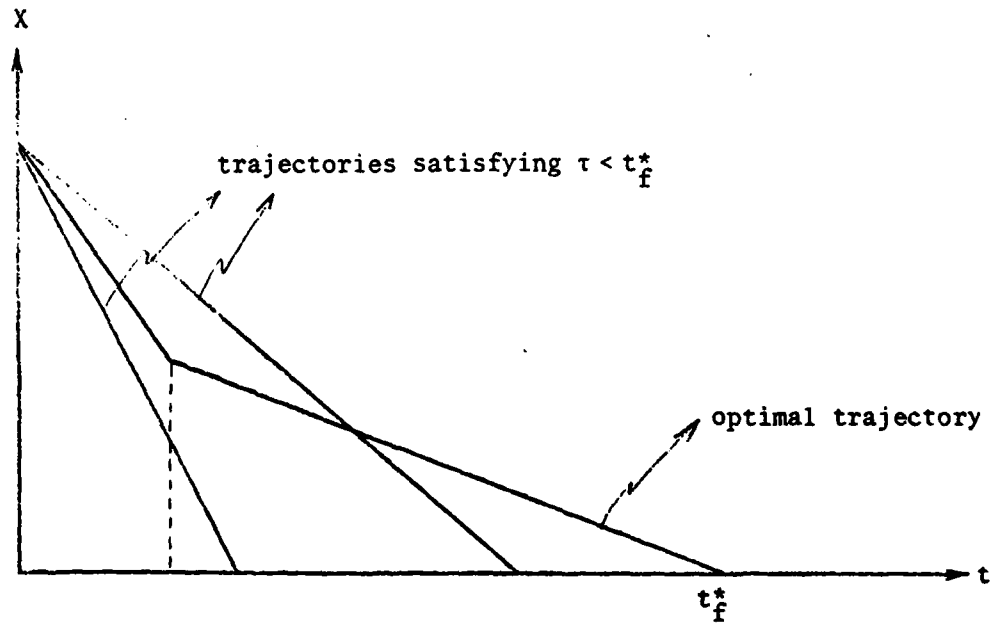


Figure 4.9: Trajectories in the Z-plane for Theorem 4.3

□ Theorem 4.3

Recall that we deal with single-destination networks with unity weightings in the cost functional.

Theorem 4.4

Let $\hat{\underline{x}}(t)$ be the one-switch-optimal-solution and t' its switch time. Then we have

$$\hat{X}(t') = X^*(t') \quad .$$

Or in words, the one-switch-optimal-solution represented in the Z-plane has its switch exactly on the optimal trajectory.

Proof

We will prove the theorem by constructing, for any one-switch solution whose switch point is not on the optimal trajectory in the Z-plane, a one-switch solution with a switch point exactly on the optimal trajectory and with a smaller delay (see Figure 4.10).

Consider an arbitrary one-switch trajectory $\hat{\underline{x}}(t)$ whose switch point is not on the optimal trajectory in the Z-plane. Then, from Theorem 4.2 follows that

$$\hat{X}(t') > X^*(t')$$

and from Theorem 4.3 follows that $\hat{t}_f \geq t_f^*$.

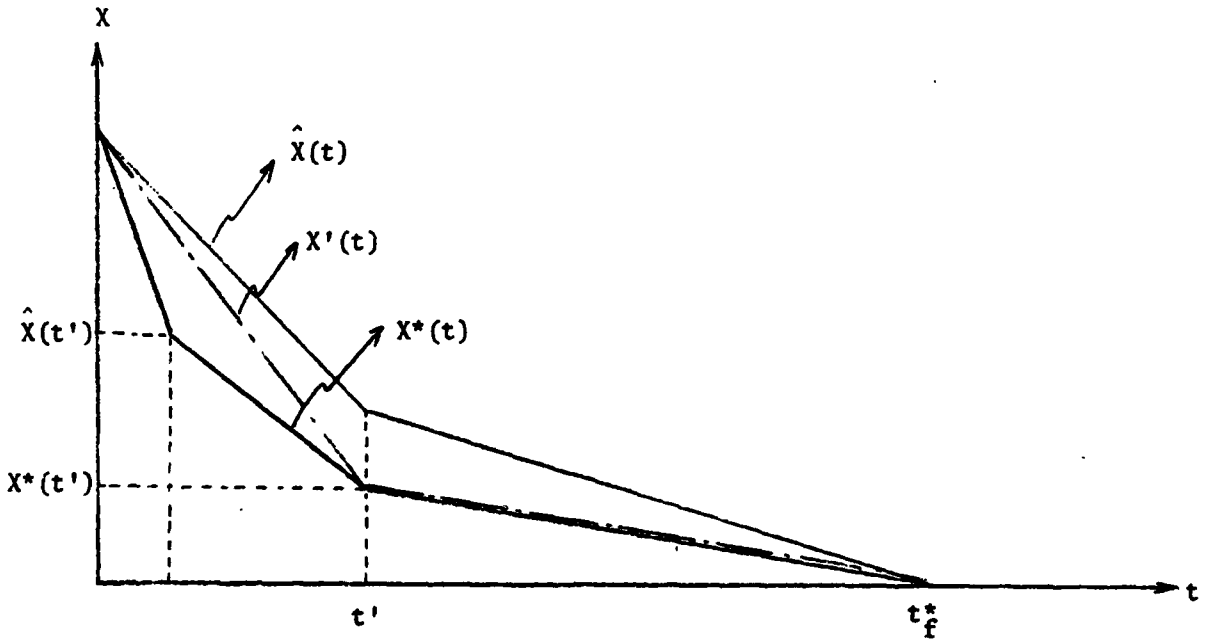


Figure 4.10: Trajectories in the Z-plane for Theorem 4.4.

We construct now a one-switch trajectory $\underline{x}'(t)$, whose switch point is on the optimal trajectory in the Z-plane, having a cost smaller than the cost of the one-switch solution considered above. Theorem 4.1 implies that there exists a non-switch trajectory taking $\underline{x}_o$ to $\underline{x}^*(t')$ in a time period t', and a trajectory taking $\underline{x}^*(t')$ to zero in a time period $t_f^* - t'$. Let $\underline{x}'(t)$ be this overall trajectory. This constructed one-switch trajectory has a cost smaller than the cost of $\hat{\underline{x}}(t)$ since they have the same switch time, $X'(t') < \hat{X}(t')$ and $t_f' = t_f^* \leq \hat{t}_f$ (see Figure 4.10).

□ Theorem 4.4

Definition 4.3

$$D \triangleq -\frac{X_o}{t_f} ,$$

is the average slope in the Z-plane.

Definition 4.4

$$\dot{X}_i \triangleq \frac{X(t_i) - X(t_{i-1})}{t_i - t_{i-1}} ,$$

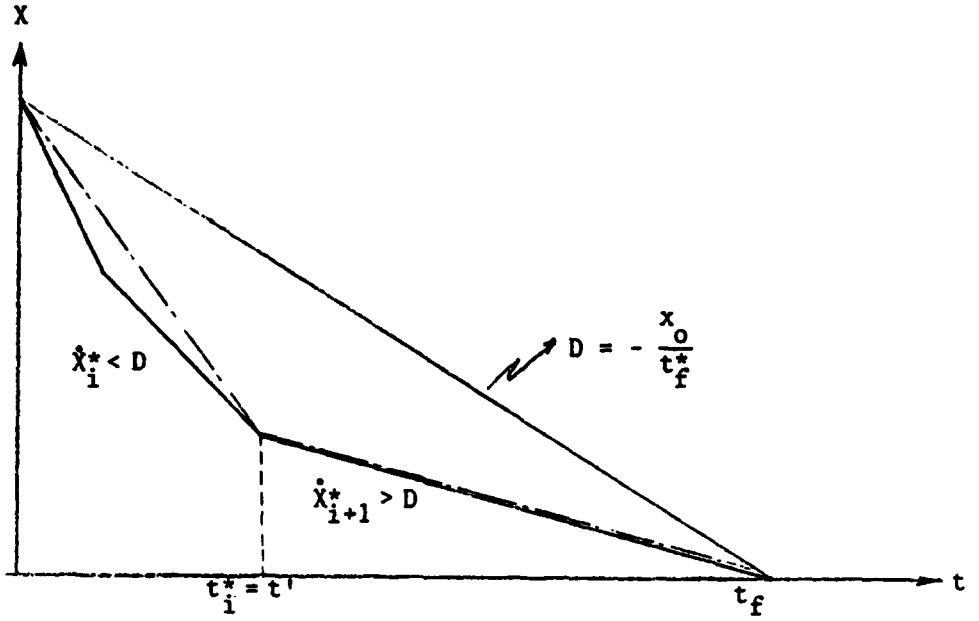
that is, $\dot{X}_i$ is the slope of a trajectory represented in the Z-plane during $[t_{i-1}, t_i]$.

Theorem 4.5

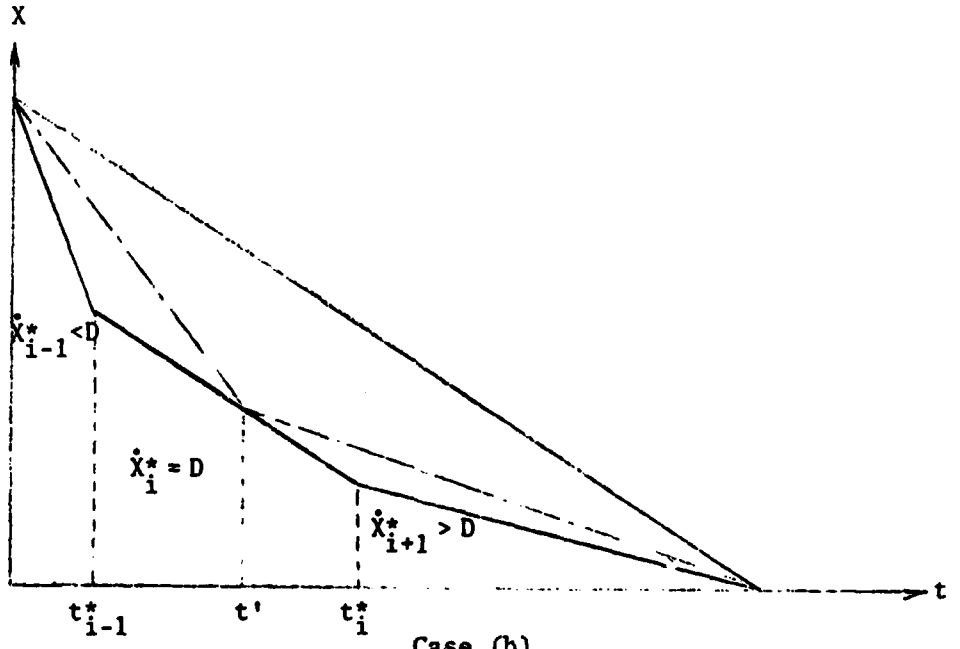
The switch time in one-switch-optimal-trajectories occurs at:

(a) a switch time t_i^* of the optimal trajectory, if $\dot{X}_i^* < D$ and $\dot{X}_{i+1}^* > D$;

(b) or at any time in the time interval $[t_{i-1}^*, t_i^*)$, if $\dot{X}_i^* = D$, $\dot{X}_{i+1}^* > D$ and $\dot{X}_{i-1}^* < D$.



Case (a)



Case (b)

Figure 4.11: The two cases of Theorem 4.5

Proof

We will find the switch time t' of the one-switch-optimal-trajectory by representing the cost function J as a sole function of t' and then, by differentiation, we will find the optimal t'.

The cost function for a one-switch trajectory will be represented as the sum of the areas of two triangles (see Figure 4.12).

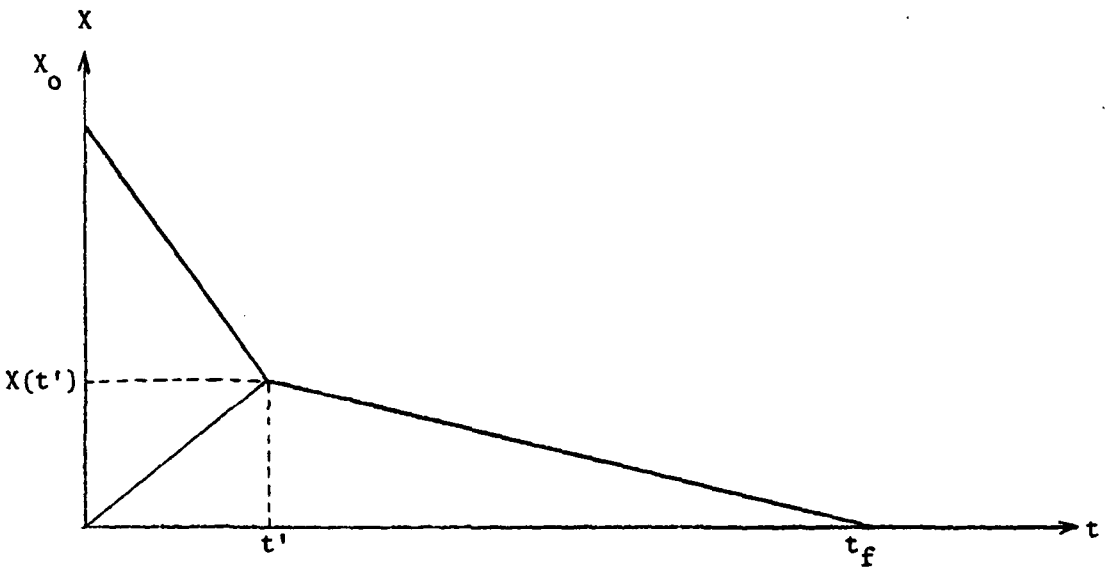


Figure 4.12: The cost J of the one-switch trajectory as the sum of the areas of two triangles.

From Figure 4.12 follows that

$$J = \frac{X_o t'}{2} + \frac{X(t')t_f}{2} \quad ,$$

For the one-switch-optimal-trajectory $t_f = t_f^*$. Then for this case

$$J = \frac{X_o t'}{2} + \frac{X(t')t_f^*}{2} \quad . \tag{4.17}$$

From (4.17) we see that J is determined by t' and $X(t')$. But in Theorem 4.4 was proved that $X(t')$ is located exactly on the optimal trajectory in the Z-plane, therefore $X(t')$ can be written as

$$X(t') = X_o + \dot{X}_1^*\tau_1^* + \dot{X}_2^*\tau_2^* + \ldots + \dot{X}_{\ell-1}^*\tau_{\ell-1}^* + \dot{X}_\ell^*(t'-t_\ell^*) \quad , \tag{4.18}$$

where $\ell - 1$ is the number of switches in the optimal solution from t_o to t', and $\dot{X}_j^*$ is the sum of the elements of the state rate vector in $[t_{j-1}^*, t_j^*)$. Then

$$J(t') = \frac{X_o t'}{2} + \frac{X_o + \sum_{j=1}^{\ell-1} \dot{X}_j^*\tau_j^* + \dot{X}_\ell^*(t'-t_\ell^*)}{2} t_f^* \quad , \quad t_{\ell-1}^* \leq t' \leq t_\ell^* \quad . \tag{4.19}$$

By differentiating (4.19) with respect to t' we get

$$\frac{dJ}{dt'} = \frac{X_o}{2} + \frac{\dot{X}_\ell^* t_f^*}{2} \quad , \tag{4.20}$$

and by replacing D (see Definition 4.3) into (4.20) we have:

$$\frac{dJ}{dt'} = (-D + \dot{X}_\ell^*) \frac{t_f^*}{2} \quad . \tag{4.21}$$

For continuing the proof we need the following proposition:

Proposition 4.2

The optimal solution satisfies

$$\dot{X}_j^* \leq \dot{X}_{j+1}^* \quad , \quad j = 1, \ldots, f-1 \quad .$$

Proof

From Proposition 2.3 follows that the optimal solution satisfies that $\dot{X}(t)$ is minimal $\forall t$ subject to the constraints $\dot{x}_i \leq 0$, $\forall x_i \in I_p$, and $\dot{x}_i = 0$, $\forall x_i \in B_p$. Since we deal with optimal solutions with piecewise constant controls satisfying $\dot{x} \leq 0$, the number of empty nodes can only be increased at switching times. Therefore some of the constraints of the type $\dot{x}_i \leq 0$ can be transformed after a switching time to constraints of the type $\dot{x}_i = 0$.

Hence, after a switching time, the constraint region can be equal to or smaller than the constraint region before the switching time, and hence $\dot{X}_j^* \leq \dot{X}_{j+1}^*$, $j = 1,\ldots,f-1$.

□ Proposition 4.2

From Proposition 4.2 follows that if $\dot{X}_\ell^* < D$, then $\dot{X}_j^* < D$, $\forall j < \ell$, and hence, from (4.21), $\frac{dJ}{dt'} < 0$, $\forall t' \in [t_o, t_\ell]$.

Now let t_i^* be the switching time of the optimal solution for which $\dot{X}_i^* < D$ and $\dot{X}_{i+1}^* > D$. Then it follows that the switching time t' of the one-switch-optimal-solution is t_i^* (see Figure 4.11, case (a)).

On the other hand, it is possible that there exists a time interval $[t_{i-1}^*, t_i^*)$ in which $\dot{X}_i^* = D$; then from Proposition 4.2 follows that $\dot{X}_{i-1}^* < D$ and $\dot{X}_{i+1}^* > D$, and the switching time t' of the one-switch-optimal-solution satisfies $t' \in [t_{i-1}^*, t_i^*)$ (see Figure 4.11, case (b).

□ Theorem 4.5

Theorem 4.6

If the one-switch-optimal-solution has no switches, then the ℓ-switches-optimal-solution $(\ell > 1)$ has no switches.

Proof

We will prove the theorem by contradiction.

Suppose that there exists an ℓ-switches-optimal trajectory with a cost smaller than the cost of the non-switch solution. From Theorem 4.3 follows that in these two solutions the final time is t_f^* (since, if the final time is greater than t_f^*, then it is possible to decrease the cost contradicting the optimality assumption).

The above fact and Proposition 4.2 imply that all switches of the ℓ-switches-optimal-trajectory are in the Z-plane underneath the non-switch-optimal-trajectory. Then, Theorem 4.1 implies that it is possible to construct a one-switch-optimal-trajectory with a smaller cost than the non-switch-trajectory, contradicting the assumption that the one-switch-optimal-solution has no switches.

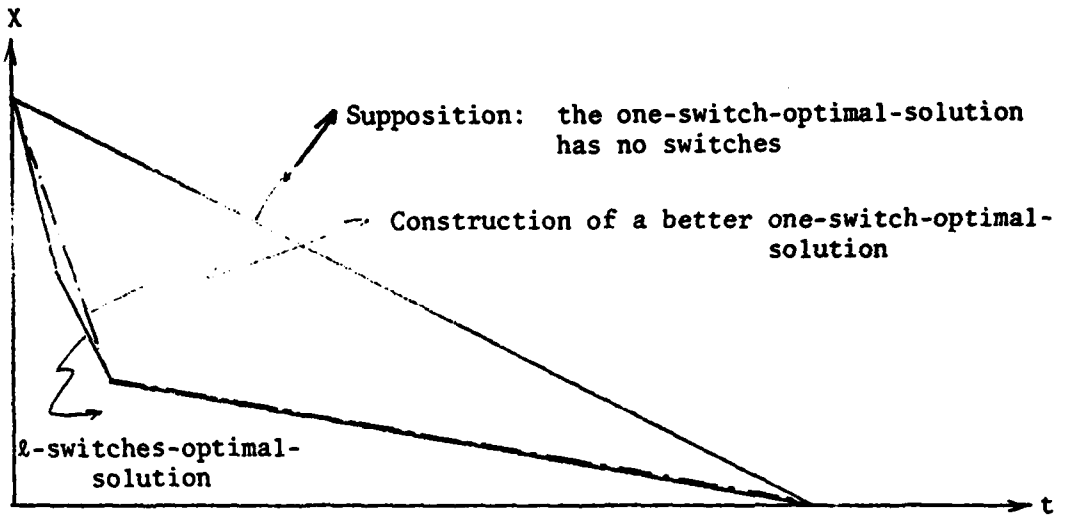


Figure 4.13: Trajectories in the Z-plane for Theorem 4.6.

4.3 Algorithm for Solving the Open-Loop Optimal Dynamic Routing Problem for Single-Destination Networks with Unity Weightings in the Cost Functional

4.3.1 The Algorithm

We present now an algorithm that solves the open-loop optimal dynamic routing problem for single-destination networks with unity weightings in the cost functional.

The proposed algorithm is based on the theory developed in Section 4.2.

We present now a brief description of the algorithm, and then we explain it in detail. At the end of this section the algorithm is presented as composed of two operations, with the second one applied repeatedly.

The algorithm finds an optimal solution by solving a series of linear programs. This is done by finding first the non-switch-optimal-solution and the final time of the optimal solution, i.e. t_f^*. Then a switch is allowed in the solution, so that the algorithm finds now the one-switch-optimal-solution. The knowledge of t_f^* allows to calculate the one-switch-optimal-solution by solving a linear program. The algorithm progresses by finding the 3, 7, 15, etc.-switches-optimal-solutions by allowing one extra-switch in each of the intervals of the previous solution until the optimal solution is found. In each step, the solutions are obtained by solving a linear program.

Explanation of the Algorithm

From Theorem 4.3 follows that the final time of the optimal solution is t_f^*, where t_f^* is the minimal final time among all feasible solutions having piecewise constant controls. This time is calculated by using Theorem 4.1 which says that for any trajectory passing from $\underline{x}_o$ to $\underline{x}_f$ in a time period t_f, there exists a non-switch trajectory passing from $\underline{x}_o$ to $\underline{x}_f$ in the same time period. Hence t_f^* will be found by looking for the non-switch trajectory having minimal final time, that is by solving the linear program:

$$t_f^* = \min t \tag{4.22a}$$

subject to

$$\left.\begin{aligned} &\underline{x}_o + B\underline{z} = 0 \\ &D\underline{z} - \underline{C}t \leq 0 \\ &\underline{z} \geq 0, \quad t \geq 0 \end{aligned}\right\} \tag{4.22b}$$

Note that the constraints (4.22b) are the constraints (3.24) - (3.27) for a non-switch trajectory.

The knowledge of t_f^* allows us to obtain the one-switch-optimal-solution by solving a linear program. In order to do so we represent the cost function, for a one-switch trajectory, as the sum of the areas of two triangles (see Figure 4.12 and equation (4.17)).

If we denote by t_1 the switch time, then $X(t_1) = -\underline{1}^T B \underline{z}_2$ (see Figure 4.14).

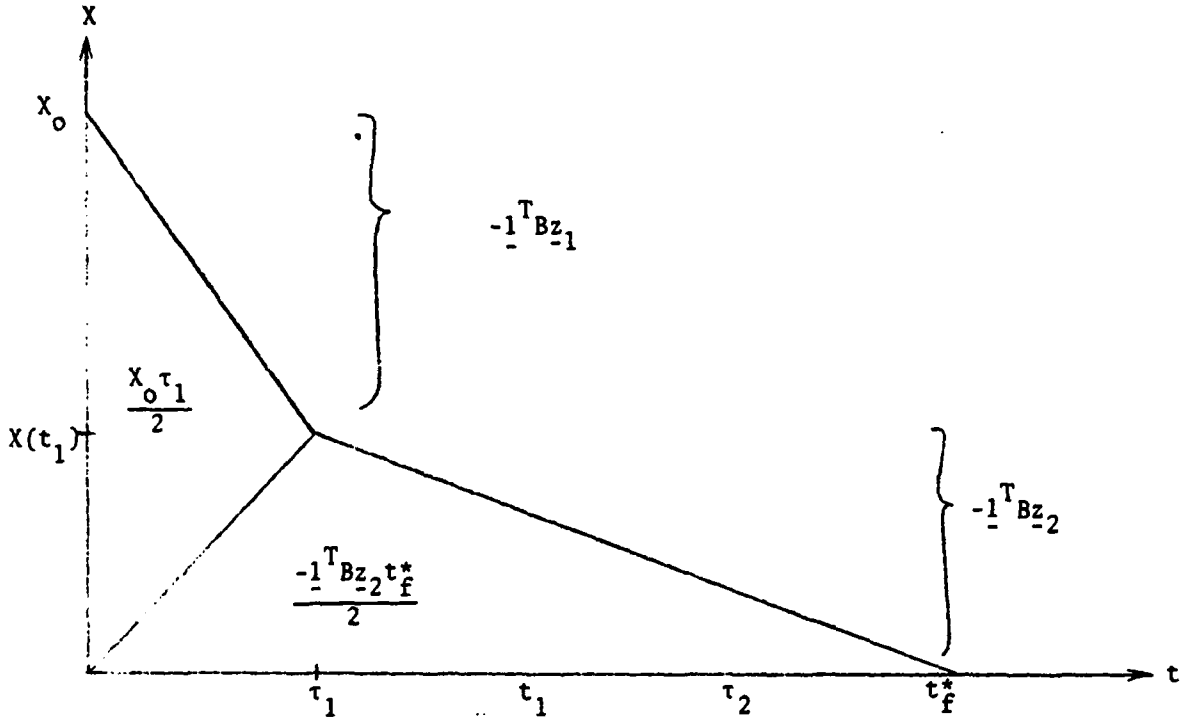


Figure 4.14: The performance index J for the one-switch-optimal-solution.

Hence, from (4.17) follows that

$$J = \frac{X_o \tau_1}{2} - \frac{\underline{1}^T B_2 t_f^*}{2} .$$

Therefore, the one-switch-optimal solution can be obtained by solving the linear program:

$$\min J = \frac{X_o \tau_1}{2} - \frac{1^T B z_2}{2} t_f^* \quad , \tag{4.23a}$$

subject to

$$\left.\begin{array}{l} \underline{x}_o + B\underline{z}_1 \geq 0 \\ \underline{x}_o + B\underline{z}_1 + B\underline{z}_2 = 0 \\ D\underline{z}_i - C\tau_i \leq 0 \\ \underline{z}_i \geq 0 \\ \tau_i \geq 0 \end{array}\right\} i = 1,2 \tag{4.23b}$$

Note that the constraint region (4.23b) is the constraint region (3.24) - (3.27) for the one-switch case.

If the optimal solution has no switches, there will be no decrease in the cost function in this step of the algorithm. If the optimal solution has exactly one switch, then it is received in this step of the algorithm and in the next step there will be no decrease in the cost function (this fact will be explained later).

From Theorem 4.4 follows that $X'(t_1')$ is equal to $X^*(t_1')$ where $\underline{x}'(t)$ is the one-switch-optimal-trajectory and t_1' its switch time. Note that $X'(t_1')$ is indeed equal to $X^*(t_1')$, but it is possible for $\underline{x}'(t_1')$ to be different from $\underline{x}^*(t_1')$. If these two vectors were equal it would have been possible to separate the problems into two problems: a problem with initial condition $\underline{x}_o$, end condition $\underline{x}'(t_1')$ and final time τ_1', and a problem with initial condition $\underline{x}'(t_1')$, end condition zero and final time τ_2'. Since this is not the case however, the problem is more complicated.

With the results of the results of the one-switch-optimal-solution as a basis, we will find next the best solution among all trajectories passing through $X'(t_1')$ and having two more switches in addition to the switch

at t_1', a switch in $[t_o,t_1']$ and another in $[t_1',t_f^*]$. This problem can be solved using linear programming if we represent the cost function as the sum of the areas appearing in Figure 4.15.

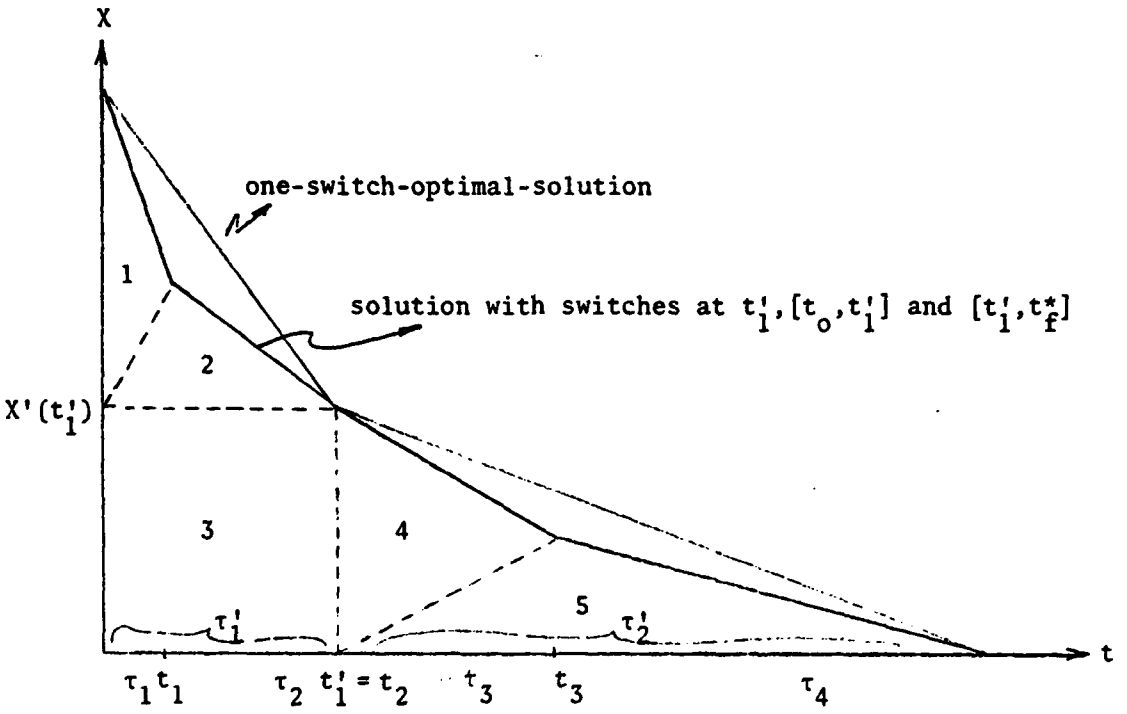


Figure 4.15: Performance index as the sum of areas for a three-switches solution.

The expression obtained is:

$$J = \underbrace{\frac{[X_o - X'(t_1')]}{2}\tau_1}_{1} \underbrace{- \frac{\underline{1}^T \underline{Bz}_2 \tau_1'}{2}}_{2} + \underbrace{X'(t_1')\tau_1'}_{3} + \underbrace{\frac{X'(t_1')\tau_3}{2}}_{4} \underbrace{- \frac{\underline{1}^T \underline{Bz}_4 \tau_2'}{2}}_{5}, \quad (4.24)$$

where the number under each expression denotes the corresponding area in Figure 4.15. All the variables refer to the three-switches trajectory and all known values (from the previous step of the algorithm) are marked with a "prime".

The constraints (3.24)-(3.27) will be written in two parts, a part before t_1' and the other from t_1' until t_f^*. In addition it is known that at t_1' the state vector $\underline{x}(t_1')$, which must be determined, satisfies

$$\sum_{i=1}^{n} x_i(t_1') = X'(t_1') \quad ,$$

and that the additional constraints $\tau_1 + \tau_2 = \tau_1'$ and $\tau_3 + \tau_4 = \tau_2'$ must hold. It follows then that the constraint region is:

$$\begin{aligned}
&\underline{x}_o + B\underline{z}_1 \geq \underline{x}(t_1') \\
&\underline{x}_o + B\underline{z}_1 + B\underline{z}_2 = \underline{x}(t_1') \\
&\sum_{i=1}^{n} x_i(t_1') = X'(t_1') \\
&\underline{x}(t_1') + B\underline{z}_3 \geq 0 \\
&\underline{x}(t_1') + B\underline{z}_3 + B\underline{z}_4 = 0 \\
&\tau_1 + \tau_2 = \tau_1' \\
&\tau_3 + \tau_4 = \tau_2' \\
&\left.\begin{aligned} D\underline{z}_i - C\underline{\tau}_i &\leq 0 \\ \underline{z}_i &\geq 0 \\ \tau_i &\geq 0 \end{aligned}\right\} i = 1,\ldots,4
\end{aligned} \tag{4.25}$$

Hence, the linear program to be solved is to minimize (4.24) subject to (4.25).

If there is a decrease in the cost function, a switch is added in each of the time intervals, a new linear program is solved, and the procedure is repeated until the optimal solution is found.

The algorithm can be described in a more concise way if we look at it as composed by two operations, as follows:

Operation 1

Find the final time of the optimal solution (t_f^*) and the non-switch optimal-solution by solving the linear program

$$t_f^* = \min t$$

subject to

$$\begin{aligned} \underline{x}_o + B\underline{z} &= 0 \\ D\underline{z} - \underline{C}t &= 0 \qquad (4.26) \\ \underline{z} &\geq 0 \\ t &\geq 0 \end{aligned}$$

Operation 2

A feasible solution $\underline{x}'(t)$ with ℓ switches, satisfying

$$X'(t_i') = X^*(t_i')$$

for all its switching times t_i', $i = 1,\ldots,\ell$, is given. Find the feasible solution with minimum cost function among all solutions having a switch in each of the time intervals $[t_i', t_{i+1}']$, $i = 0,\ldots,\ell$, $(t_{\ell+1}' = t_f^*)$, in addition to the switches t_i', $i = 1,\ldots,\ell$, and satisfying $X'(t_i')$ equal to $X^*(t_i')$, $i = 1,\ldots,\ell$. This solution will have $2\ell + 1$ switches and is obtained by solving the linear program (see Figure 4.16):

$$\min J = \sum_{i=o}^{\ell} \left\{ \frac{X'(t_i') - X'(t_{i+1}')}{2} \tau_{2i+1} - \frac{\underline{1}^T B \underline{z}_{2i+2} \tau_{i+1}'}{2} \right\} + \sum_{i=1}^{\ell} X'(t_i')\tau_i', \qquad (4.27a)$$

subject to:

$$
\begin{aligned}
&\underline{x}_o + B\underline{z}_1 \geq \underline{x}(t_1') \\
&\underline{x}_o + B\underline{z}_1 + B\underline{z}_2 = \underline{x}(t_1') \\
&\sum_{i=1}^{n} x_i(t_1') = X'(t_1') \\
&\underline{x}(t_1') + B\underline{z}_3 \geq \underline{x}(t_2') \\
&\underline{x}(t_1') + B\underline{z}_3 + B\underline{z}_4 = \underline{x}(t_2') \\
&\sum_{i=1}^{n} x_i(t_2') = X'(t_2') \\
&\quad \cdots \\
&\underline{x}(t_{\ell-1}') + B\underline{z}_{2\ell-1} \geq \underline{x}(t_\ell') \\
&\underline{x}(t_{\ell-1}') + B\underline{z}_{2\ell-1} + B\underline{z}_{2\ell} = \underline{x}(t_\ell') \\
&\sum_{i=1}^{n} x_i(t_\ell') = X'(t_\ell') \qquad (4.27b) \\
&\underline{x}(t_\ell') + B\underline{z}_{2\ell+1} \geq 0 \\
&\underline{x}(t_\ell') + B\underline{z}_{2\ell+1} + B\underline{z}_{2\ell+2} = 0 \\
&\tau_1 + \tau_2 = \tau_1' \\
&\tau_3 + \tau_4 = \tau_2' \\
&\quad \cdots \\
&\tau_{2\ell+1} + \tau_{2\ell+2} = \tau_{\ell+1}' \\
&\left.\begin{aligned} D\underline{z}_i - \underline{C}\tau_i &\leq 0 \\ \underline{z}_i &\geq 0 \\ \tau_i &\geq 0 \end{aligned}\right\} i = 1,\ldots,2\ell+2
\end{aligned}
$$

Note that $\sum_{i=1}^{\ell} X'(t_i')\tau_i'$ is constant (is received from the previous step of the algorithm) and appears in J only for $\ell \geq 1$.

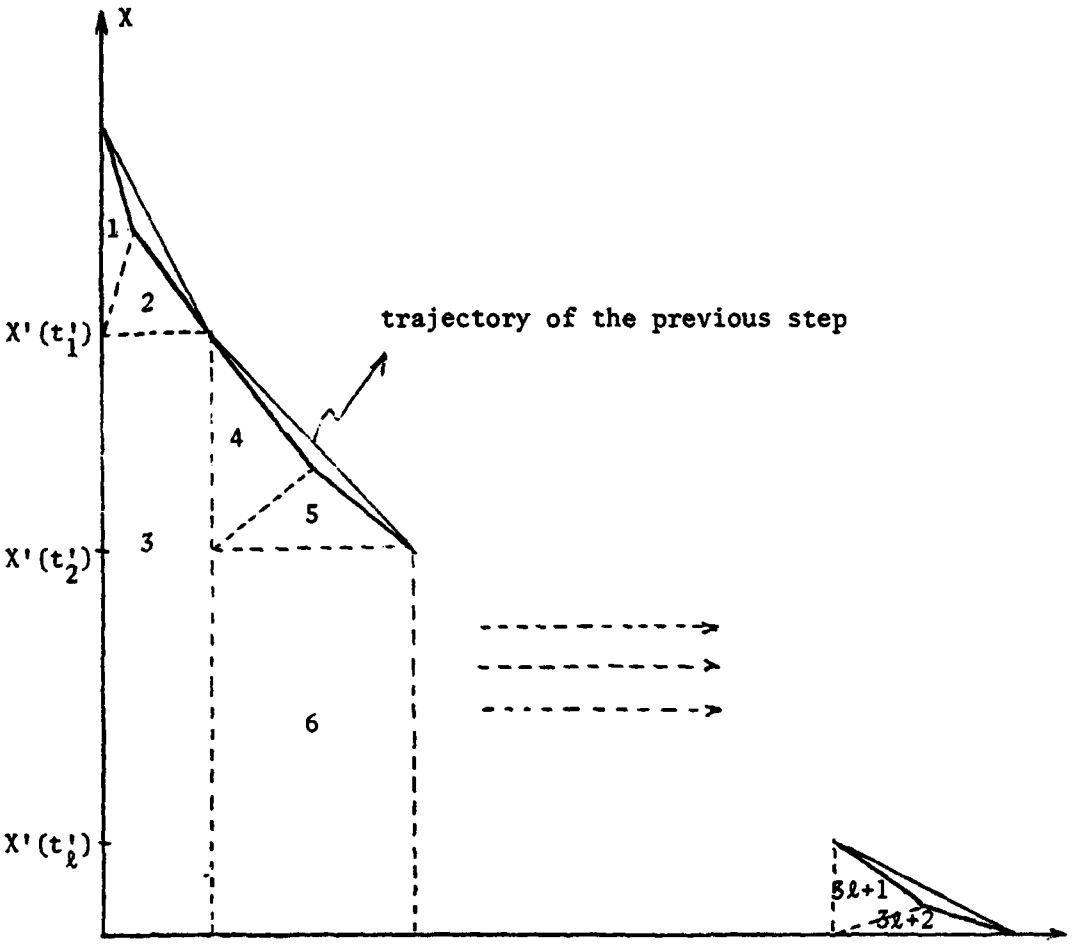


Figure 4.16: J as the sum of areas for Operation 2.

These two operations are applied as follows:

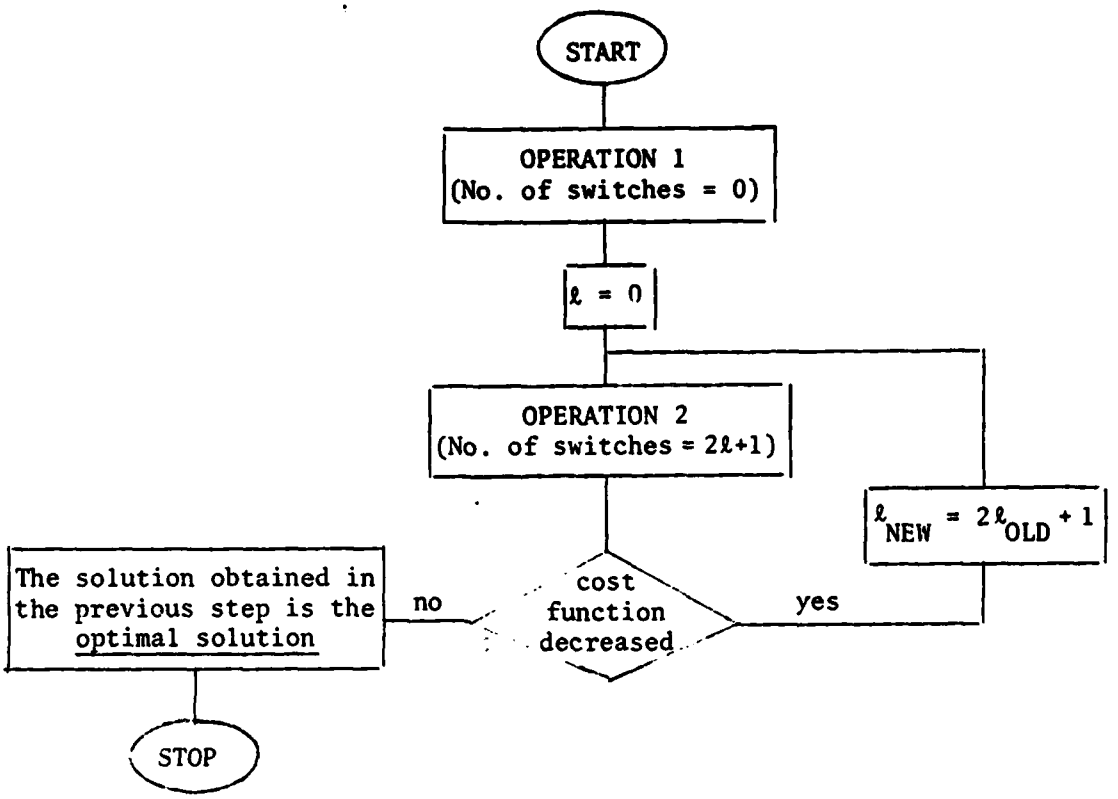


Figure 4.17: The two operations of the algorithm.

Next we shall prove that the algorithm converges to the optimum in a finite number of steps. In order to do so we shall prove first the following propositions:

Proposition 4.3

Given that:

$$X'(t'_j) = X^*(t'_j), \quad j = 1,\ldots,\ell \quad ,$$

where $\underline{x}'(t)$ is the solution of Operation 2 with ℓ switches. Then

$$X(t_i) = X^*(t_i), \quad i = 1,\ldots,2\ell+1 \quad ,$$

where $\underline{x}(t)$ is the solution of the next step of the algorithm, i.e. $\underline{x}(t)$ is the solution of Operation 2 with $2\ell+1$ switches.

Proof

From Theorem 4.2 follows that $X(t_i) \geq X^*(t_i)$, $i = 1,\ldots,2\ell+1$. Hence, if the proposition is not satisfied, $X(t_i) > X^*(t_i)$ for at least one switching time.

We now assume that there exists a solution to Operation 2 which does not satisfy the proposition, and then construct a solution to Operation 2 that does satisfy the proposition, having a smaller cost function.

The constructed solution (see Figure 4.18) is the solution passing through $\underline{x}^*(t_i)$ for all $i = 1,\ldots,2\ell+1$. It is clear that for this solution $X(t_i)$ is equal to $X^*(t_i)$ at all switching times, and its cost is smaller than the cost of the assumed solution.

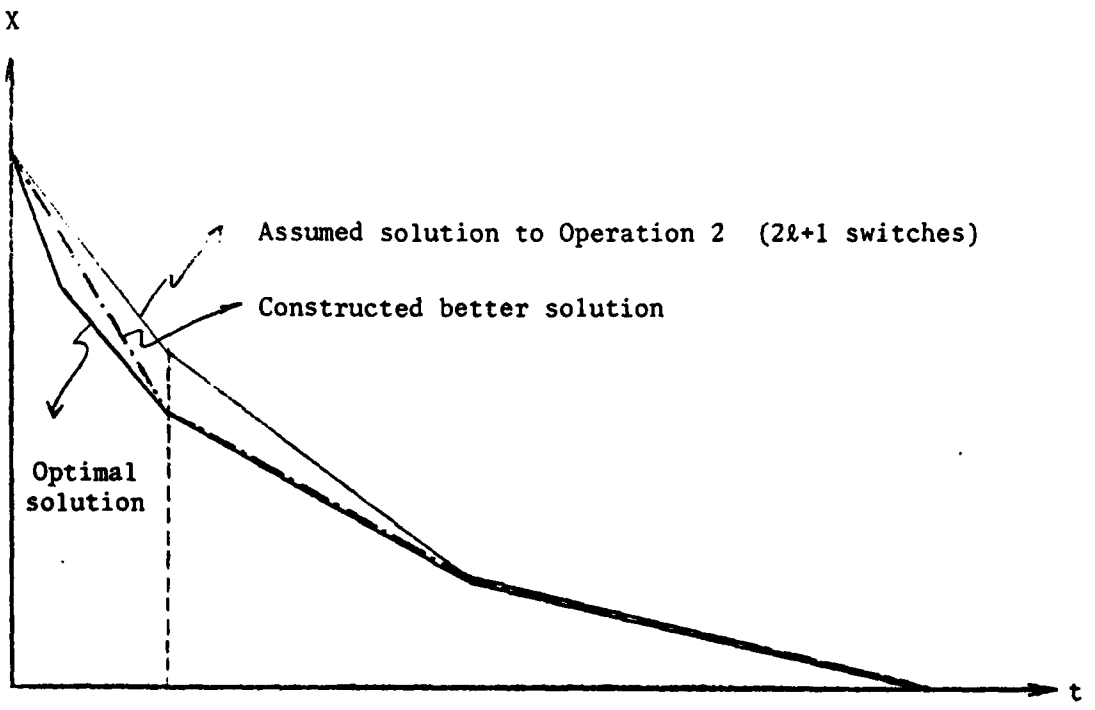


Figure 4.18: Trajectories in the Z-plane for Proposition 4.3

□ Proposition 4.3

Proposition 4.4

If after a step of the algorithm there is no decrease in the cost function, then further steps (allowing more switches in each time interval) will not decrease the cost function.

Proof

A step of the algorithm means allowing a switch in each of the time intervals $[t_i', t_{i+1}']$, $i = 0,\ldots,\ell$ where t_i', $i = 1,\ldots,\ell$ are the switching times obtained in the previous step of the algorithm. Suppose then, that the present step does not decrease the cost function value, and look at the problems defined by the switching times, that is, look at the problems with initial condition $\underline{x}(t_i')$, end condition $\underline{x}(t_{i+1}')$, and final time τ_i', $i = 1,\ldots,\ell+1$. From Theorem 4.6 follows that allowing more switches in each of these problems will not decrease the cost function. Hence, further steps will not decrease the cost function.

□ Proposition 4.4

Theorem 4.7: Convergence Theorem

The proposed algorithm converges to the optimal solution in a finite number of steps.

Proof

The optimal solution has a finite number of switches distributed in $[t_o, t_f^*]$. In each step of the algorithm the allowed number of switches increases, in such a way that in a finite number of steps we will obtain a solution with the required number of switches distributed in $[t_o, t_f^*]$, like the optimal solution. From Proposition 4.3 follows that the solution of Operation 2 for this case satisfies $X(t_i)$ equal to $X^*(t_i)$, and hence it is the optimal solution.

□ Theorem 4.7

The theorem above implies that after a finite number of steps an optimal solution will be achieved. From Proposition 4.4 follows that the stopping criterion of the algorithm is no decrease of the cost function value after some step; that is, if in a certain step there is no decrease in the cost function value, the optimal solution was reached in the previous step.

The algorithm proposed in this section was programmed (see Appendix C), and several networks were solved. The program has the property that it discovers when the time between switches is zero, and for this case it does not allow more switches in this interval. In this way unnecessary new variables are defined in the following steps of the algorithm.

In the next section we present some of the networks that were solved.

4.3.2 Examples

In this section we present numerical results obtained for the solution of the optimal open-loop routing in a number of networks. The problems were solved using a computation program written in FORTRAN (see Appendix C). The program was run on an IBM 370/168 computer and for each example, besides Example 4.3, a maximal memory region of 512K was used.

We present now three examples. In the first we present the network parameters, the results of the different steps of the algorithm and the optimal solution. In the other two, only the network parameters and the optimal solution are printed.

Example 4.1

Refer to the network appearing in Figure 4.19 with initial condition $\underset{\sim}{x}_o = (2,5,4)^T$. The numbers on the links are their capacities.

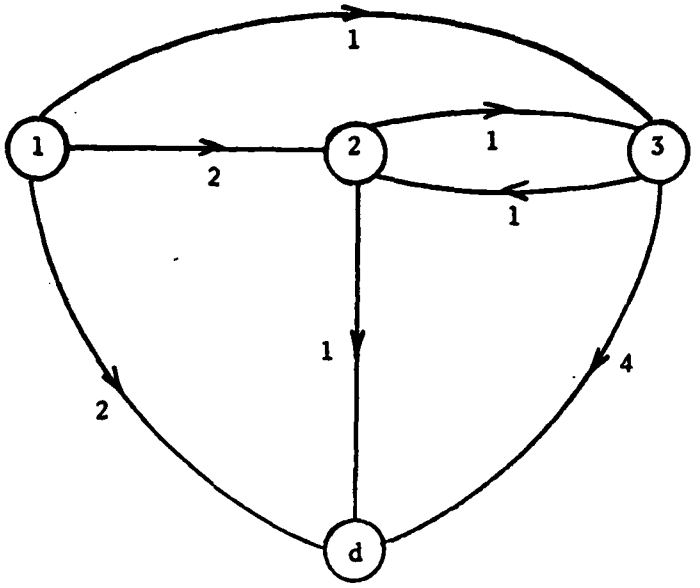


Figure 4.19: Network for Example 4.1

The results of the program are:

```
THE CAPACITY VECTOR IS:

2.000  1.000  2.000  1.000  1.000  1.000  4.000


THE B MATRIX IS:

-1.0-1.0-1.0 0.0 0.0 0.0 0.0

 1.0 0.0 0.0-1.0-1.0 1.0 0.0

 0.0 1.0 0.0 1.0 0.0-1.0-1.0

THE INITIAL CONDITION IS:
 2.00  5.00  4.00
        : OPTIMUM COST VALUE     2.49999RE 00
        : ERROR FLAGS     0 AND  2.980232E-07
        : OPTIMUM SOLUTION
   1  0.0                2  0.0                3  1.99999RE 00
   4  2.499995E 00       5  2.500006E 00       6  0.0
   7  6.499996E 00       8  2.499999E 00


 THE TOTAL TIME IS:0.2499999F 01
        : OPTIMUM COST VALUE     1.025000E 01
        : ERROR FLAGS    -1 AND  4.758372E-06
        : OPTIMUM SOLUTION
   1  0.0                2  0.0                3  2.000000E 00
   4  1.333333F 00       5  1.333333E 00       6  0.0
   7  5.333332E 00       8  0.0                9  0.0
  10  0.0               11  1.166667E 00      12  1.166667E 00
  13  0.0               14  1.166668E 00      15  1.333333E 00
  16  1.166667E 00      17 -1.192093E-06
        : OPTIMUM COST VALUE     6.805552E 00
        : ERROR FLAGS    -1 AND  1.847744E-06
        : OPTIMUM SOLUTION
   1  0.0                2  0.0                3  2.000000E 00
   4  1.000000E 00       5  1.000000E 00       6  0.0
   7  4.000000E 00       8  0.0                9  0.0
  10  0.0               11  3.333335E-01      12  3.333335E-01
  13  0.0               14  1.333331E 00      15  0.0
  16  0.0               17  0.0               18  5.960464E-07
  19  5.960464E-07      20  0.0               21  2.543130E-06
  22  0.0               23  0.0               24  0.0
  25  1.166667E 00      26  1.166667E 00      27  0.0
  28  1.166667E 00      29  0.0               30  2.333333F 00
  31  1.053015E-06      32  1.000000E 00      33  3.333333E-01
  34  5.960464E-07      35  1.166667E 00      36  0.0
  37  8.145968E-07
```

: OPTIMUM COST VALUE 5.138857E 00

: ERROR FLAGS -1 AND 2.574921E-05

: OPTIMUM SOLUTION

1	0.0	2	0.0	3	1.999998E 00
4	9.999996E-01	5	9.999996E-01	6	0.0
7	3.999990E 00	8	0.0	9	0.0
10	0.0	11	9.536743E-07	12	9.536743E-07
13	0.0	14	6.675720E-06	15	0.0
16	0.0	17	1.788139E-06	18	3.333327E-01
19	3.333327E-01	20	0.0	21	1.333328E 00
22	0.0	23	0.0	24	0.0
25	7.947273E-08	26	7.947273E-08	27	0.0
28	7.947273E-08	29	0.0	30	0.0
31	0.0	32	5.960464E-07	33	5.960464E-07
34	0.0	35	0.0	36	0.0
37	0.0	38	0.0	39	0.0
40	1.907348E-06	41	0.0	42	0.0
43	0.0	44	0.0	45	0.0
46	9.139381E-07	47	9.139381E-07	48	0.0
49	2.106030E-06	50	0.0	51	0.0
52	0.0	53	1.166664E 00	54	1.166664E 00
55	0.0	56	1.166664E 00	57	1.788139E-06
58	2.999999E 00	59	9.999966E-01	60	0.0
61	2.333333E 00	62	1.192356E-06	63	0.0
64	2.333330E 00	65	1.788403E-06	66	9.999996E-01
67	9.536743E-07	68	3.333327E-01	69	7.947273E-08
70	5.960464E-07	71	0.0	72	9.139381E-07
73	1.166664E 00	74	0.0	75	0.0
76	0.0	77	2.264976E-06		

TAU(1)=0.1000000E 01

U(1)=0.0

U(2)=0.0

U(3)=0.2000000E 01

U(4)=0.1000000E 01

U(5)=0.1000000E 01

U(6)=0.0

U(7)=0.4000000E 01

TAU(2)=0.3333335E 00

U(1)=0.0

U(2)=0.0

U(3)=0.0

U(4)=0.1000000E 01

U(5)=0.1000000E 01

U(6)=0.0

U(7)=0.3999991E 01

TAU(3)=0.1166667E 01

U(1)=0.0

U(2)=0.0

U(3)=0.0

U(4)=0.1000000E 01

U(5)=0.1000000E 01

U(6)=0.0

U(7)=0.1000000E 01

In the first part we print the network parameters, i.e. the capacity vector, the B-matrix and the initial conditions. The first element of the capacity vector is the capacity of the link that corresponds to the first column of the B-matrix, and so on. In the second part the results of the different steps of the algorithm are presented (the meaning of the different variables of the linear programs appears in Appendix C). Note that in this example, there is no elimination of variables when a time interval is zero in order to get an easy reference to the meaning of the variables. The steps of the algorithm are:

(a) Operation 1: In the first linear program, the final time (TOTAL TIME) is calculated, and the non-switch-optimal-solution is obtained.

(b) Operation 2 for $\ell = 0$: The one-switch-optimal-solution is calculated. Since there is an improvement in the cost, this solution replaces the non-switch-optimal-solution and we continue with the algorithm.

(c) Operation 2 for $\ell = 1$: A switch is allowed in each of the intervals between switches. Since there is no further improvement in the cost function, the solution for $\ell = 0$ is the optimal solution. For each of the intervals we print the control vector $\underline{u}$. Note that the control vector $\underline{u}$ is printed in the same order as the capacities and the columns of B.

From the optimal solution and from matrix B we conclude that in the first interval $\dot{x}_1 = -2$, $\dot{x}_2 = -2$ and $\dot{x}_3 = -3$; in the second interval $\dot{x}_1 = 0$, $\dot{x}_2 = -2$ and $\dot{x}_3 = -3$; and in the third interval $\dot{x}_1 = 0$, $\dot{x}_2 = -2$ and $\dot{x}_3 = 0$ (see Figure 4.20).

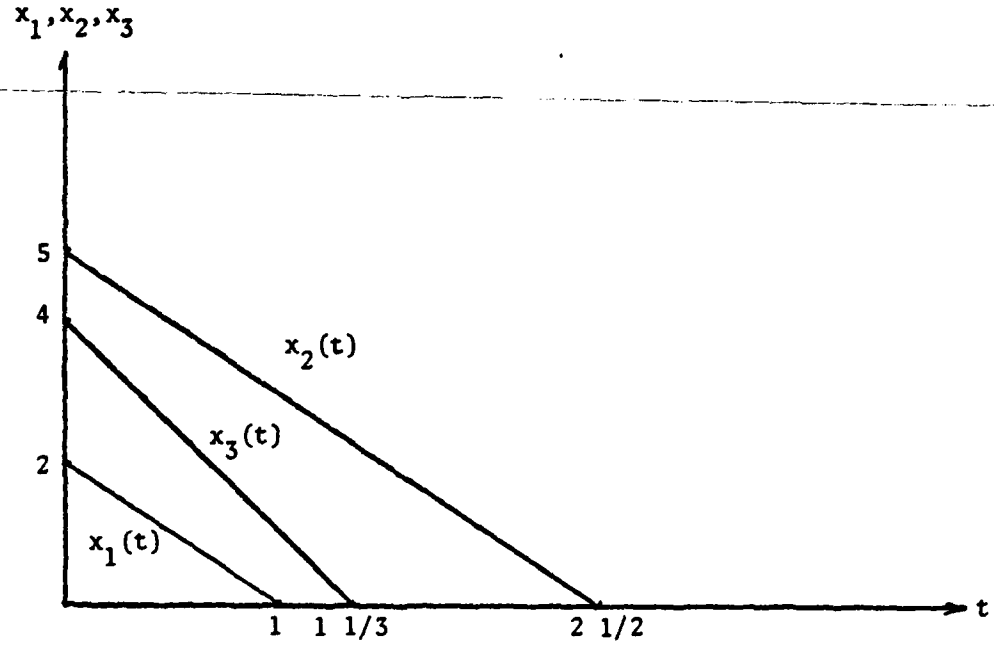


Figure 4.20: State trajectories for Example 4.1

□ Example 4.1

Example 4.2

Refer to the network appearing in Figure 4.21 with the initial condition $\underline{x}_o = (1,4,21,15,20)^T$.

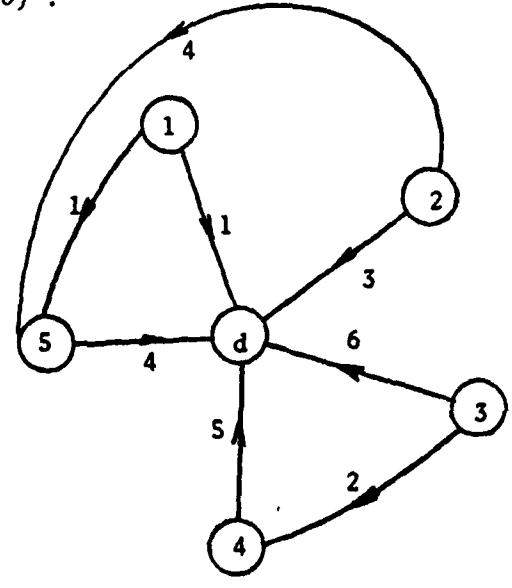


Figure 4.21: Network for Example 4.2

The results of the program are:

```
THE CAPACITY VECTOR IS:

1.000  1.000  4.000  3.000  6.000  2.000  5.000  4.000


THE B MATRIX IS:

-1.0-1.0  0.0  0.0  0.0  0.0  0.0  0.0
 0.0  0.0-1.0-1.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0-1.0  1.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0-1.0-1.0  0.0
 1.0  0.0  1.0  0.0  0.0  0.0  0.0-1.0

THE INITIAL CONDITIUN IS:
 1.00  4.00 21.00 15.00 20.00


     TAU( 1)=0.1000000E 01

   U( 1)=0.0

   U( 2)=0.1000000E 01

   U( 3)=0.0

   U( 4)=0.3000000E 01

   U( 5)=0.6000000E 01

   U( 6)=0.0

   U( 7)=0.5000000E 01

   U( 8)=0.4000000E 01
```

```
   TAU( 2)=0.3333321E 00

U( 1)=0.0

U( 2)=0.0

U( 3)=0.0

U( 4)=0.3000000E 01

U( 5)=0.6000000E 01

U( 6)=0.0

U( 7)=0.5000001E 01

U( 8)=0.4000000E 01


   TAU( 3)=0.1666667E 01

U( 1)=0.0

U( 2)=0.0

U( 3)=0.0

U( 4)=0.0

U( 5)=0.6000000E 01

U( 6)=0.0

U( 7)=0.5000000E 01

U( 8)=0.4000000E 01
```

TAU(4)=0.5000000E 00

U(1)=0.0

U(2)=0.0

U(3)=0.0

U(4)=0.0

U(5)=0.6000000E 01

U(6)=0.0

U(7)=0.0

U(8)=0.4000002E 01

TAU(5)=0.1500000E 01

U(1)=0.0

U(2)=0.0

U(3)=0.0

U(4)=0.0

U(5)=0.0

U(6)=0.0

U(7)=0.0

U(8)=0.4000000E 01

From the optimal solution and matrix B we conclude that (see Figure 4.22):

first interval: $\dot{x}_1 = -1,\ \dot{x}_2 = -3,\ \dot{x}_3 = -6,\ \dot{x}_4 = -5,\ \dot{x}_5 = -4,$

second interval: $\dot{x}_1 = 0,\ \dot{x}_2 = -3,\ \dot{x}_3 = -6,\ \dot{x}_4 = -5,\ \dot{x}_5 = -4,$

third interval: $\dot{x}_1 = 0,\ \dot{x}_2 = 0,\ \dot{x}_3 = -6,\ \dot{x}_4 = -5,\ \dot{x}_5 \quad -4,$

fourth interval: $\dot{x}_1 = 0,\ \dot{x}_2 = 0,\ \dot{x}_3 = -6,\ \dot{x}_4 = 0,\ \dot{x}_5 = -4,$

fifth interval: $\dot{x}_1 = 0,\ \dot{x}_2 = 0,\ \dot{x}_3 = 0,\ \dot{x}_4 = 0,\ \dot{x}_5 = -4.$

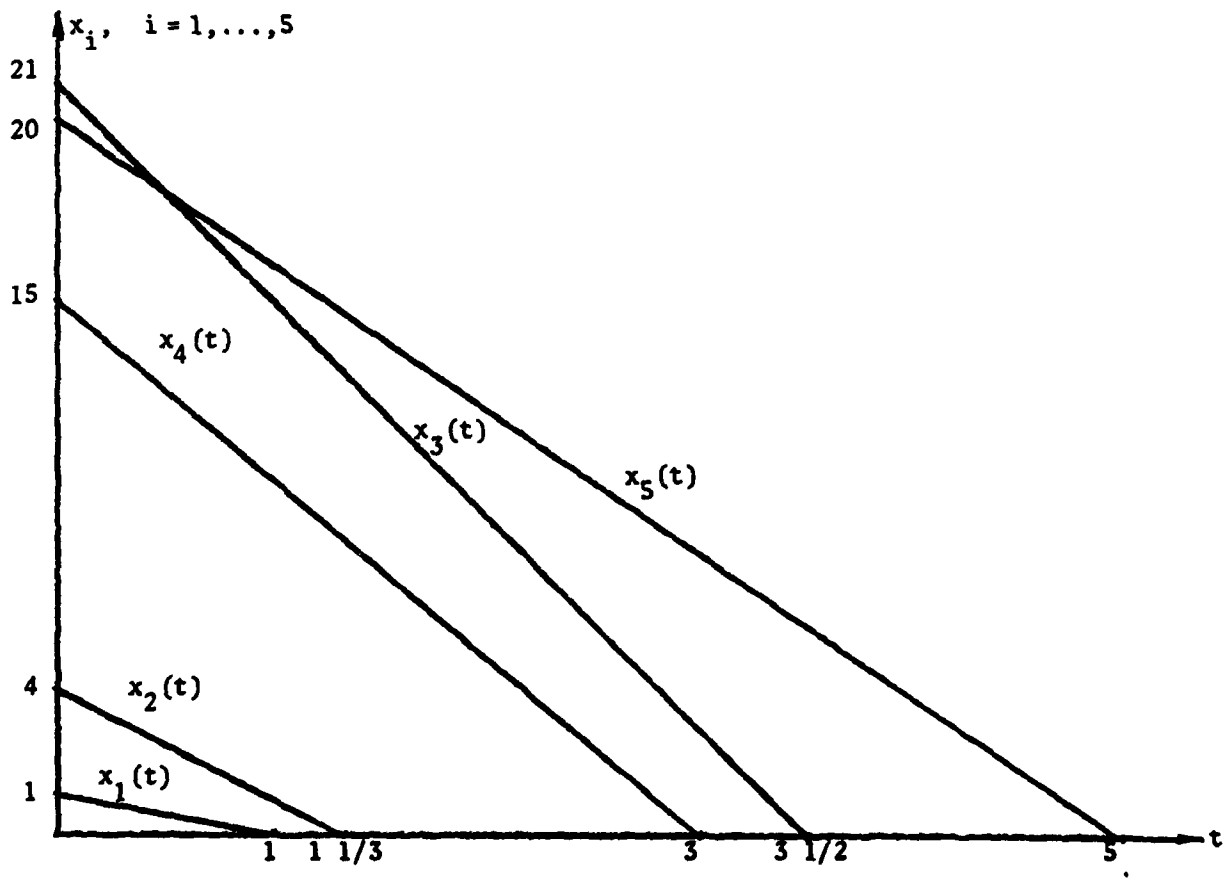


Figure 4.22: State trajectories for Example 4.2

□ Example 4.2

Example 4.3

Refer to the network appearing in Figure 4.23 with the initial condition $\underset{-o}{x} = (7,3,1,5,4,4,3)^T$.

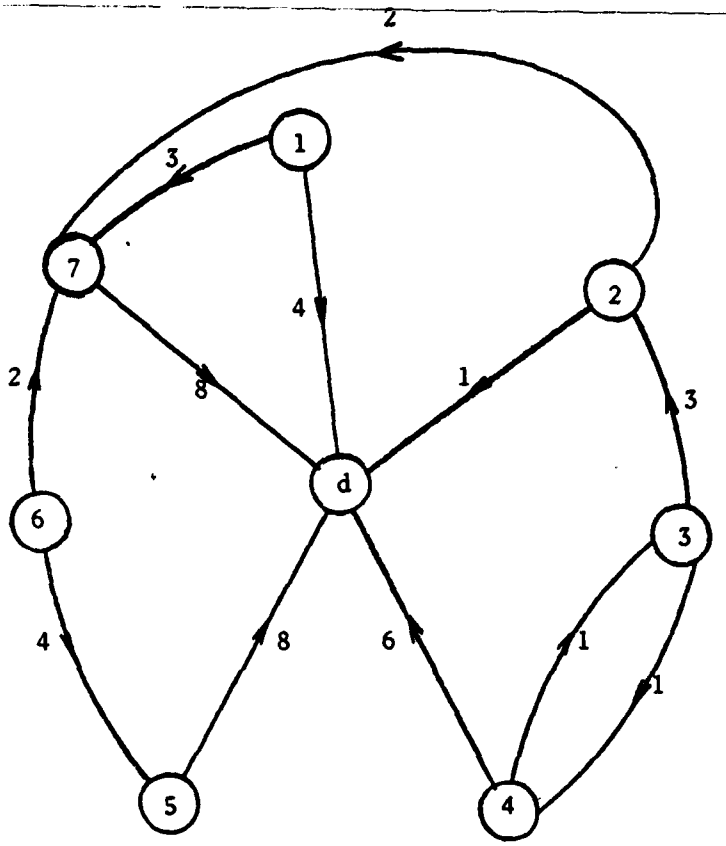


Figure 4.23: Network for Example 4.3

The results of the program are:

```
THE CAPACITY VECTOR IS:

3.000  4.000  2.000  1.000  3.000  1.000  1.000  6.000  8.000

4.000  2.000  8.000

THE B MATRIX IS:

-1.0-1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

 0.0 0.0-1.0-1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

 0.0 0.0 0.0 0.0-1.0-1.0 1.0 0.0 0.0 0.0 0.0 0.0

 0.0 0.0 0.0 0.0 0.0 1.0-1.0-1.0 0.0 0.0 0.0 0.0

 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0-1.0 1.0 0.0 0.0

 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0-1.0-1.0 0.0

 1.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0-1.0

THE INITIAL CONDITION IS:
 7.00  3.00  1.00  5.00  4.00  4.00  3.00
```

NC SWITCHS IN THE OPTIMAL SOLUTION

TAU(1)=0.1000000E 01

U(1)=0.2999998E 01

U(2)=0.4000004E 01

U(3)=0.1999998E 01

U(4)=0.1000000E 01

U(5)=0.0

U(6)=0.1000000E 01

U(7)=0.0

U(8)=0.6000000E 01

U(9)=0.8000006E 01

U(10)=0.4000004E 01

U(11)=-.3814697E-06

U(12)=0.7999999E 01

From the optimal solution and matrix B we conclude that (see Figure 4.24):

$$\dot{x}_1 = -7, \quad \dot{x}_2 = -3, \quad \dot{x}_3 = -1, \quad \dot{x}_4 = -5, \quad \dot{x}_5 = -4, \quad \dot{x}_6 = -4, \quad \dot{x}_7 = -3 .$$

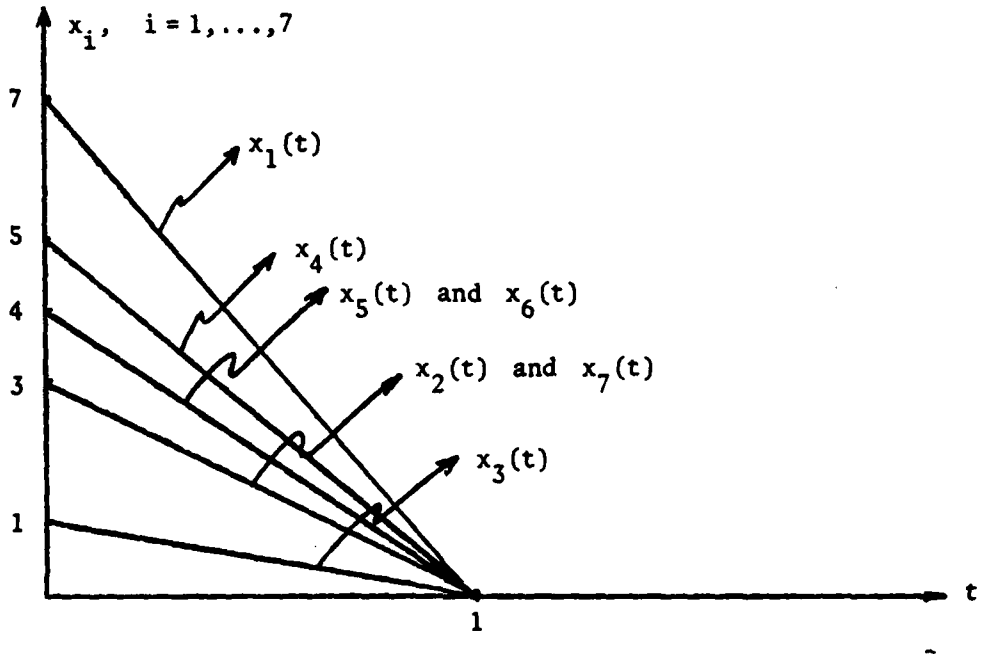


Figure 4.24: State trajectories for Example 4.3

□ Example 4.3

In this work, a library computation program for solving linear programs was used. This program is not very appropriate for solving our problem. For small networks with a small number of switches in the optimal solution, only a few seconds of CPU were required for obtaining the optimal solution. But for larger networks with a large number of switches in the optimal solution, tens and even hundreds of seconds of CPU were required. For this reason larger networks could not be completely solved. Further research is needed for obtaining an efficient linear program that will permit the solution of large networks.

Chapter 5

Open-Loop Solutions for the Dynamic Routing Problem by Existing Nonconvex Quadratic Programming Methods

5.1 Introduction

In this chapter we shall propose other approaches for solving the open-loop dynamic routing problem.

In Chapter 3, the open-loop dynamic routing problem was presented as a nonconvex quadratic program (Problem 3.2), and it was proved that this problem is not concave, quasi-concave, quasi-convex and pseudo-convex (see Section 3.3.3). In light of this fact, it is possible to solve Problem 3.2 by using existing methods for nonconvex quadratic programming.

In this chapter we will indicate the most appropriate methods with their advantages and disadvantages.

All the methods mentioned in this chapter and others related to nonconvex quadratic programming are overviewed in Appendix B.

5.2 Ritter's Algorithm

This algorithm finds a global minimum of a nonconvex quadratic program (see Method (19) in Appendix B). It is a finite and relatively easily implementable algorithm for finding local minima. But, in spite of Ritter's convergence proof in a finite number of steps to a global minimum, Zwart found an example that can not be solved in a finite number of steps. Therefore, in order to solve Problem 3.2 using Ritter's algorithm, we must test convergence for this specific case.

5.3 The Cabot-Francis Algorithm

Cabot and Francis developed an algorithm for solving the nonconvex quadratic programming for the cases in which it is previously known that some extreme point of the constraint region is an optimal point. Therefore, in order to use this algorithm for the solution of Problem 3.2, it must be proved that the above mentioned condition holds. For this purpose we need the following lemma:

Lemma 5.1

For single destination networks with unity weightings in the cost functional, there exists an optimal solution to the dynamic routing problem, such that:

(a) For every time interval $[t_{i-1},t_i)$, $i \in F$, the optimal control is an extreme point of

$$\left.\begin{aligned} \underline{D}\underline{u}_i^* &\leq \underline{C}\,, && (5.1a)\\ \underline{u}_i^* &\geq 0\,, && (5.1b)\\ \underline{b}_j^T\underline{u}_i^* &= 0\,, \quad \forall x_j \in I_i && (5.1c)\\ \underline{b}_\ell^T\underline{u}_i^* &= 0\,, \quad \forall x_\ell \in B_i && (5.1d) \end{aligned}\right\} \qquad (5.1)$$

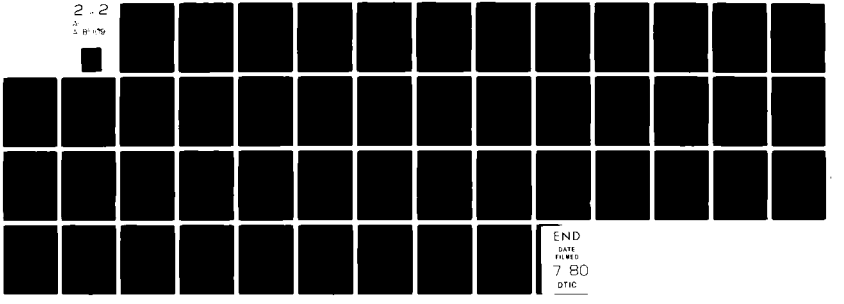
(b) The number of switches is less than or equal to $n-1$, i.e., $f \leq n$.

Proof

From (2.20) and Proposition 2.2, it is known that the optimal control on the time interval $[t_{i-1},t_i)$, $i \in F$ is obtained from the linear program:

AD-A085 109 MASSACHUSETTS INST OF TECH CAMBRIDGE LAB FOR INFORMA--ETC F/G 12/2
OPEN-LOOP SOLUTIONS FOR THE DYNAMIC ROUTING PROBLEM.(U)
MAY 80 S SHATS, A SEGAL N00014-75-C-1183
UNCLASSIFIED LIDS-R-992 NL

2.2



END
DATE FILMED
7 80
DTIC

$$\min \underline{\lambda}^T B \underline{u}_i \quad , \tag{5.2a}$$

subject to

$$\left.\begin{aligned} &\underline{u}_i \in U \quad , \\ &\dot{x}_j \leq 0 \quad , \quad \forall x_j \in I_i \quad , \\ &\dot{x}_\ell = 0 \quad , \quad \forall x_\ell \in B_i \quad . \end{aligned}\right\} \tag{5.2b}$$

From the linear program above follows that there exists an extreme point of (5.2b) that is an optimal solution to (5.2). Hence, from (2.5) and (2.7), follows that this optimal solution is an extreme point of (5.1).

In order to obtain an upper bound for the minimum number of switches that can be obtained in an optimal solution satisfying (5.1), we shall look for an optimal trajectory with maximum number of switches. In order to do so, we assume that all components of $\underline{x}_o$ are greater than zero. Starting from t_o we apply a control that satisfies part (a) of the lemma. From Property 2.2 we know that it is optimal to continue with this control until a set of state variables become zero, or until $\underline{x}^*(t)$ reaches the origin. For finding the upper bound to the minimum number of switches, we assume that only one state variable becomes zero. From this moment we apply a new control until another state variable becomes zero. If we continue in this way we will arrive at the origin with maximum $n - 1$ switches. Then:

$$f - 1 \leq n - 1$$

or

$$f \leq n$$

□ Lemma 5.1

This lemma allows us to prove Theorem 5.1 and using it to show that Problem 3.2 has the property required by the Cabot-Francis method.

The constraint region of Problem 3.2 ((3.24) - (3.27)) is presented in (5.3) in a different way because we want to associate to each constraint

of the type (5.1d) a slack variable equal to zero and to each constraint of the type (5.1c) a possibly nonzero slack variable (see (3.26) and (5.3c)). Moreover, the constraints $\tau_i \geq 0$ are not written in (5.3) because from (3.24) and (3.25) follows that they are redundant.

Theorem 5.1

In single destination networks with unity weightings in the cost functional, there exists an optimal solution to Problem 3.2 at an extreme point of the constraint region:

$$\left.\begin{array}{lll} \underline{z}_i \geq 0 , & & (5.3a) \\ D\underline{z}_i - C\underline{\tau}_i \leq 0 , & i \in F & (5.3b) \\ \underline{x}_o + B\underline{z}_1 - \underline{k}_1 = 0 , & & \\ \underline{x}_o + \sum_{i=1}^{p} B\underline{z}_i - \underline{k}_1 + \sum_{j=2}^{p} \underline{k}_j = 0 , \quad p = 2,3,\ldots,f & & (5.3c) \\ \underline{x}_o + \sum_{i=1}^{f} B\underline{z}_i = 0 , & & (5.3d) \end{array}\right\} \quad (5.3)$$

Proof

All the extreme points of a linear constraint region are basic solutions of the system of equations defining it. And each basic solution is an extreme point. Therefore, a point is an extreme point if and only if the number of nonzero variables is less than or equal to the number of constraints defining the region. The theorem will therefore be proved if we show that there exists an optimal solution, satisfying Lemma 5.1, that has a number of nonzero variables in (5.3) less than or equal to the number of constraints defining (5.3).

We denote by V the constraint region of all control variables $\underline{u}_1, \underline{u}_2, \ldots, \underline{u}_f$ (formed by f regions of the type (5.1)) and define:

$\alpha \triangleq$ {the number of equations of the type (5.1a) and (5.1b) in V} ,

$\beta \triangleq$ {the number of nonzero variables (including slack variables) in the equations of the type (5.1a) and (5.1b) in V} .

By using these definitions we shall count the number of nonzero variables and the numbers of equations in V for an optimal solution.

The number of equations of the type (5.1c) in V is nf.

If we denote by r the number of equations of the type $\underline{b}_\ell^T \underline{u}_i^* = 0$, $i \in F$, the number of nonzero variables (not including the variables $\underline{u}_i^*$, $i \in F$) in the equations of the type (5.1c) are $nf - r$. Note that all those variables are slack variables.

From the above mentioned calculations it is clear that for an optimal solution satisfying Lemma 5.1, the number of equations in V is $\underline{nf + \alpha}$ and the number of nonzero variables in V is $\underline{\beta + nf - r}$.

Since the optimal solution in question is an extreme point of V (Lemma 5.1), the number of nonzero variables is less than or equal to the number of equations. Therefore

$$\left\{\begin{matrix}\text{number of nonzero} \\ \text{variables in } V\end{matrix}\right\} = \beta + nf - r \leq nf + \alpha = \left\{\begin{matrix}\text{number of} \\ \text{equations in } V\end{matrix}\right\} ,$$

which implies

$$\beta - r \leq \alpha . \tag{5.4}$$

We now prove the theorem, by counting the number of nonzero variables and equations in (5.3) for an optimal solution satisfying Lemma 5.1.

The number of equations of the type (5.3a) and (5.3b) is equal to the number of equations of the type (5.1a) and (5.1b) (i.e. α), since they are the same equations multiplied by τ_i, $i \in F$. Likewise there are $n(f-1)$ equations in (5.3c) and n equations in (5.3d). Therefore, the number of equations in (5.3) is $\underline{nf + \alpha}$.

In (5.3a) and (5.3b) there are $\beta + f$ nonzero variables; the f time periods τ_i, $i \in F$ (these are nonzero since $\tau_i = 0$ means that there is no time interval $[t_{i-1}, t_i)$) and the β nonzero variables of equations (5.1a) and (5.1b) (that appear in (5.3) multiplied by the respective τ_i).

In (5.3c) there are $n(f-1) - r$ additional nonzero variables, that is all the slack variables, apart from the r slack variables associated with the constraints $\underline{b}_{\ell-i}^T \underline{u}_i^* = 0$, $i \in F$.

From the above mentioned facts it is clear that the number of nonzero variables in (5.3) is $\underline{\beta + f + n(f-1) - r}$.

From Lemma 5.1 and (5.4) it is known that

$$\beta - r \leq \alpha$$

and

$$f \leq n \ ,$$

therefore

$$\left\{\begin{matrix}\text{number of nonzero} \\ \text{variables in (5.3)}\end{matrix}\right\} = \beta + f + n(f-1) - r \leq nf + \alpha = \left\{\begin{matrix}\text{number of} \\ \text{equations in (5.3)}\end{matrix}\right\}.$$

Consequently, an optimal solution satisfying Lemma 5.1 is an extreme point of (5.3).

□ Theorem 5.1

In Theorem 5.1 it was proved that there exists an optimal solution to Problem 3.2 that is an extreme point of (5.3). Therefore the Cabot-Francis method can be used for solving the problem. That is, the problem

can be solved by ranking the extreme points of (5.3) (see Method (22) in Appendix B).

The most serious limitation of this method is that all the representations of some of the extreme points of (5.3) are needed and as a result of that the efficiency of the algorithm is not good for degenerate problems (Problem 3.2 can be very degenerate).

5.4 Other Methods

Besides the previously mentioned methods, there are other methods for solving nonconvex quadratic programs that may be used for the solution of Problem 3.2. Some of those methods are: the Majthag, Whinston and Coffman method for finding a local minimum (see Method (15) in Appendix B) and the Candler-Townsley method for finding a global minimum (see Method (16) in Appendix B). The limitations of these methods are that, in the first only a local minimum is reached, and in the second there is no proof of convergence.

Other methods, relatively new, that assure convergence to a global minimum are: the generalized polars method (see Method (17) in Appendix B) and the Tone method (see Method (18) in Appendix B).

Chapter 6

Conclusions

6.1 Discussion

In this work we made use of the fact that the optimal solution is of the bang-bang type, in order to formulate the open-loop routing problem as dependent on new variables, $\underline{u}_i$ and τ_i, $i \in F$. The problem obtained was shown to be a cubic optimization problem having a nonconvex constraint region. Since this type of problems is quite difficult, the fact that all nonlinear terms include a term of the form $\underline{u}_i \tau_i$ was used for transforming the problem into a quadratic program. This new problem was shown to be nonconvex, with nonconcave, nonquasi-concave, nonquasi-convex and nonpseudo-convex cost function, and therefore it does not possess any special property that will allow an easier solution than a general nonconvex quadratic program.

Consequently two approaches were proposed for the solution of the problem. The first one based on existing methods from nonconvex quadratic programming, the second consisting of developing a special algorithm. This special algorithm was developed for single destination networks with unity weightings in the cost functional.

The algorithm reduces the nonconvex quadratic program to a series of linear programs. The number of switches in the optimal solution depends on the initial condition. In some cases the number of switches may be small and one obtains relatively small linear programs. For other initial conditions, specially in big networks, the optimal solution may have many switches (up to n-1), and then the solution of large linear programs may be required in the last steps of the algorithm.

In this work we only test the algorithm, using a general-purpose, linear program from a library of computation programs, thus excluding the possibility of testing large problems.

Further research is required in order to develop special linear programs that are adequate for solving large problems.

6.2 Suggestions for Further Work

Some suggestions for further research are:

(a) The use of a special-purpose linear program, instead of the general one used here, for obtaining a more efficient algorithm that will permit solving of large networks.

(b) All the results of this work refer to networks without inputs, but they can be extended to networks with constant inputs.

(c) The algorithm proposed in Chapter 4 was developed for single-destination networks with unity weightings in the cost functional; further research is needed for extending the results to multi-destination networks with priorities in the cost functional.

(d) An interesting direction for further investigation may be to develop an algorithm based on Ritter's method, that solves the open-loop dynamic routing problem. This approach may be interesting because Ritter's method can be applied to general networks with priorities in the cost functional.

Appendix A

Properties of Functions

A.1 Convexity and Concavity of Functions

Definition A.1

A set S is said to be convex if and only if

$$\forall \underline{x},\underline{y} \in S \text{ then } (1-\lambda)\underline{x} + \lambda\underline{y} \in S \text{ for } \lambda \in [0,1] .$$

□ Definition A.1

Definition A.2

A real-valued function ϕ is said to be convex on S if:

(a) S is convex ,

(b) $\forall \underline{x},\underline{y} \in S$ then $(1-\lambda)\phi(\underline{x}) + \lambda\phi(\underline{y}) \geq \phi[(1-\lambda)\underline{x} + \lambda\underline{y}]$ for $\lambda \in [0,1]$.

□ Definition A.2

Definition A.3

A real-valued function ϕ is said to be concave on A if

(a) S is convex ,

(b) $\forall \underline{x},\underline{y} \in S$ then $(1-\lambda)\phi(\underline{x}) + \lambda\phi(y) \leq \phi[(1-\lambda)\underline{x} + \lambda\underline{y}]$ for $\lambda \in [0,1]$.

□ Definition A.3

These properties are important in mathematical programming since for a convex function the Kuhn-Tucker necessary conditions for minimum are also sufficient and every local minimum is a global minimum. Likewise, if the function ϕ is concave on S the minimum is achieved at an extreme point of S.

We now define some properties, weaker than convexity and concavity, that are sufficient for the above statements to hold.

A.2 Quasi-Convexity and Quasi-Concavity of Functions

Definition A.4

A real-valued function ϕ defined on a convex set S is said to be quasi-convex on S if and only if the set

$$\{\underline{x} \in S / \phi(\underline{x}) \leq w\} \text{ is convex for all } w \in \mathbf{R} .$$

□ Definition A.4

This property is partially responsible for the theoretical usefulness of quasi-convex functions as constraints in a nonlinear program. Although these functions need not be convex, they still determine a convex constraint set.

An example of a quasi-convex function that is not convex on the interval $[0,1]$ is

$$\phi(x) = \begin{cases} c_1 & \text{if } 0 \leq x \leq \frac{1}{2} \\ c_2 & \text{if } \frac{1}{2} < x \leq 1 \end{cases} ,$$

where c_1 and c_2 are scalars, and $c_1 \neq c_2$.

Definition A.5

A real-valued function ϕ defined on a convex set S is said to be quasi-convex on S if and only if

$$\forall \underline{x}, \underline{y} \in S, \quad \phi[(1-\lambda)\underline{x} + \lambda \underline{y}] \leq \max[\phi(\underline{x}), \phi(\underline{y})] \quad \text{for} \quad \lambda \in [0,1] .$$

□ Definition A.5

For differentiable functions, the following definition is also equivalent:

Definition A.6

A differentiable function ϕ is quasi-convex on the convex set S if and only if

$$\forall \underline{x},\underline{y} \in S, \quad \phi(\underline{y}) \leq \phi(\underline{x}) \quad \text{implies} \quad \nabla\phi(\underline{x})(\underline{y}-\underline{x}) \leq 0 \ .$$

□ Definition A.6

The concept of quasi-convex functions was introduced by De Finetti [A8], and subsequently developed by Nikaido [A13], and by Arrow and Enthoven [A1].

Definition A.7

A real-valued function ϕ defined on a convex set S is said to be quasi-concave on S if and only if the set

$$\{\underline{x} \in S/\phi(\underline{x}) \geq w\} \quad \text{is convex for all} \quad w \in \mathbb{R} \ .$$

□ Definition A.7

Definition A.8

A real-valued function ϕ defined on a convex set S is said to be quasi-concave on S if and only if

$$\forall \underline{x},\underline{y} \in S, \quad \phi[(1-\lambda)\underline{x} + \lambda\underline{y}] \geq \min[\phi(\underline{x}),\phi(\underline{y})] \quad \text{for} \quad \lambda \in [0,1] \ .$$

□ Definition A.8

Definition A.9

A differentiable function ϕ is quasi-concave on the convex set S if and only if

$$\forall \underline{x},\underline{y} \in S, \quad \phi(\underline{y}) \leq \phi(\underline{x}) \quad \text{implies} \quad \nabla\phi(\underline{y})(\underline{x}-\underline{y}) \geq 0 \ .$$

□ Definition A.9

An important property of quasi-concave functions is that the minimum of the function ϕ is achieved at an extreme point of the convex set S. This fact can be easily proved by noting that a quasi-concave function takes on only higher values on convex combinations than it takes on at the defining point with the smallest function value.

A.3 Pseudo-Convexity of Functions

Definition A.10

A real-valued differentiable function ϕ on a convex set S is pseudo-convex if and only if

$$\forall \underline{x},\underline{y} \in S, \quad \nabla\phi(\underline{x})(\underline{y}-\underline{x}) \geq 0 \quad \text{implies} \quad \phi(\underline{y}) \geq \phi(\underline{x}) .$$

□ Definition A.10

Every local minimum of a pseudo-convex function is global, and the Kuhn-Tucker necessary conditions are also sufficient for a local (global) minimum in a nonlinear program whose objective function is pseudo-convex and the constraints are defined by quasiconvex functions. See [A2].

This concept was first introduced by Tui [A19] and later developed by Mangasarian [A9], [A10]. See also Ponstein [A14].

Pseudo-convexity is defined by infinitely many inequalities which are in general hard to verify. For particular functions, such as products, quadratic or cubic functions, more practical criteria are available. (See [A7], [A10], [A15], [A16], [A17] and [A18].

It can easily be shown that:

Proposition A.1 (See [A9] and [A14])

A differentiable convex function is pseudo-convex and a pseudo-convex function on a convex set is quasi-convex.

□ Proposition A.1

For further reading on the convexity of quadratic functions see [A4], [A5], [A6], and on the quasi-convexity and pseudo-convexity of quadratic functions see [A3], [A7], [A11], [A12] and [A18].

Appendix B

Nonconvex Quadratic Programming

B.1 Introduction

B.1.1 Definition of a Quadratic Program

A quadratic program can be defined as follows:

$$\left.\begin{aligned} &\min\ z = \underline{c}^T\underline{x} + \underline{x}^T D\underline{x} \\ &\text{subject to} \\ &\qquad A\underline{x} = \underline{b} \\ &\text{and} \\ &\qquad \underline{x} \geq 0 \end{aligned}\right\}, \tag{B.1}$$

where $\underline{x}$ and $\underline{c}$ are column vectors of dimension n, A is an $m \times n$ matrix, $\underline{b}$ is a column vector of dimension m, and D is a symmetric $n \times n$ matrix.

Note that all the different kinds of quadratic programming problems can be written in the form (B.1) by using the following transformations:

1. It is possible to find the maximum of z instead of the minimum by using
$$\max z = \min -z \ .$$
2. It is possible to transform inequality constraints into equality constraints by adding slack variables.
3. It is possible to transform a set of ℓ free variables x_j', $j \in J$ into $\ell + 1$ nonnegative variables by using the transformations:
$$x_j' = x_j - x_o, \quad j \in J \ ,$$
where $x_j \geq 0$, $j \in J$, and $x_o \geq 0$.

B.1.2 Synopsis of the Appendix

In Section B.2 we present a review of the methods for solving the convex quadratic program, i.e. the methods for solving problem (B.1) when D is positive definite or positive semidefinite (see [A4], [A5] and [A6]).

In Section B.3 we present a review of some of the existing methods for solving nonconvex quadratic programs. This will be done by classifying the methods into four groups according to the properties of the cost function. The four groups are:

1. Methods for quasi-convex or pseudo-convex cost function (see B.3.1).
2. Methods for quasi-concave or concave cost function (see B.3.2).
3. General methods (z without special properties) (see B.3.3).
4. Special methods (see B.3.4).

For each one of these groups we will review some of the existing methods.

Since the problem to be solved (Problem 3.2) has properties that permit us to solve it by using Ritter's algorithm (see Section B.3.3), or the Cabot-Francis algorithm (see Section B.3.4), we shall explain these methods in depth.

B.2 Convex Quadratic Programming

Kuhn-Tucker's Theorem implies that a necessary condition for solving (B.1) is (see [B13], p. 213):

$$\left.\begin{aligned} \underline{v} - A^T\underline{\lambda} - 2D\underline{x} &= \underline{c} \\ x_j v_j &= 0, \quad j = 1,2,\ldots,n \\ A\underline{x} &= \underline{b}\ , \\ \underline{x} \geq 0, \quad \underline{v} &> 0 \end{aligned}\right\} . \qquad (B.2)$$

In convex quadratic programs, each stationary point is a global minimum, therefore the conditions (B.2) are also sufficient optimality conditions. Almost all methods for solving convex quadratic programs are based on the solution of this system of equations.

Some of the most important methods for solving convex quadratic programs are:

(1) Wolfe's Algorithm (see [B38]; and [B30], pp. 224-242)

This algorithm is based upon Kuhn-Tucker conditions and the SIMPLEX method for solving linear programs. A solution to (B.2) is achieved by "SIMPLEX-like" steps with the additional requirement that $x_j v_j = 0$ in each step. This condition is known as the complementarity condition.

(2) Beale's Method (see [B2]; and [B30], pp. 242-258)

Beale's method is based (like Method (1)) on the Kuhn-Tucker conditions and the SIMPLEX method, but in this method the constraint region changes during the application of the algorithm.

(3) Lemke's Method (see [B21]; and [B30], pp. 258-273)

This method is based on the Kuhn-Tucker conditions. Since D^{-1} is needed, it can be used only when D is positive definite.

(4) Theil - van de Panne's Method (see [B34])

This method consists of finding an optimal solution to the problem without constraints and then adding the constraints step by step, where in each step the optimum cost is calculated. D^{-1} is needed for the calculations, therefore this method can be used only when D is positive definite.

(5) Capacity Methods (see [B13], pp. 236-238; and [B16])

These methods are for problems with constraints of the type $D\underline{x} \leq \underline{C}$ One of them is the Houthakker method that is based on the introduction of a parametric constraint to the original constraints.

As for Methods (3) and (4), D must be positive definite.

(6) "Principle Pivot" Method (see [B8])

The principle pivot method is a method for solving the linear complementarity problem

$$\left.\begin{aligned} \underline{w} &= \underline{q} + M\underline{z} \\ \underline{w}^T\underline{z} &= 0 \\ \underline{w} \geq 0, \quad \underline{z} &\geq 0 \end{aligned}\right\} \qquad (B.3)$$

The Kuhn-Tucker conditions are a special case of (B.3); therefore, it is possible to solve (B.2) by using methods for solving (B.3).

In [B14], Hefley compared Methods (1) and (6), and his conclusion was that the principle pivot method as appears in [B8] is superior.

For other methods see [B10], [B12], [B15], [B18], [B26] and [B29].

B.3 Nonconvex Quadratic Programming

B.3.1 Special Methods for Solving Quadratic Programs with Pseudo-Convex or Quasi-Convex Cost Functions

From Section A.3 it is known that the Kuhn-Tucker conditions are necessary and sufficient for minimum, when the cost function is pseudo-convex. Therefore, it is possible to solve the nonconvex quadratic programs with pseudo-convex cost functions by using some of the methods of Section B.2.

We now present a theorem relating pseudo-convex with quasi-convex functions:

Theorem B.1 (see [A3])

If the quadratic function z (see (B.1)) is not convex but is quasi-convex in the nonnegative orthant; then, if $\underline{c} \neq 0$, the function z is pseudo-convex in the nonnegative orthant.

□ Theorem B.1

Likewise, in [A12] it was proved that when $\underline{c} = 0$, z is pseudo-convex in the nonnegative orthant only when z is convex.

From Proposition A.1, from Theorem B.1 and from the above follows that pseudo-convex and quasi-convex functions are closely related and almost all the quadratic quasi-convex functions are also pseudo-convex. Therefore the same methods can be used for solving both types of problems. For the case where the cost function is quasi-convex and $\underline{c} = 0$, a method for solving pseudo-convex problems can be used if a small perturbation on $\underline{c}$ is performed.

Martos in [A11], and Mylander in [B25] reviewed the methods for solving the pseudo-convex quadratic programming problem and concluded that the best methods are:

- ~~Beale's method~~ (see Method (2))
- Ritter's method (see Method (19) in Section B.3.3)
- (7) Keller's Method (see [B18])

  This is a variation of Method (6).
- (8) Mylander's Method (see [B26])

  This method is strongly related to Lemke's method (see Method (3)).

B.3.2 Special Methods for Solving Quadratic Programs with Concave or Quasi-Concave Cost Functions

In Sections A.1 and A.2 it was shown that if the cost function is concave or *quasi-concave*, the *minimum* of the function is located at an extreme point of the linear constraint region. All methods for solving these problems are based on this property.

Some of the most important methods for solving optimization problems with concave cost function (not necessarily quadratic) are:

(9) Tui's Method (see [B37])

Tui was one of the first investigators in dealing with this problem. In his work he proposes a method based on the extension of the cost function (which is defined on a specific region) over R^n. In each step a new constraint is created (Tui cut) or the cost function is changed.

(10) Hu's Method (see [B17])

This method is based on basical elements from Tui's work and on the SIMPLEX method. During the steps of the algorithm the constraint region changes since the method requires the addition of constraints.

(11) Zwart's Method (see [B39])

This algorithm is similar to Tui's algorithm. A proof of convergence is provided for the case when a little deviation from the optimum is permitted and, in spite of the fact that there is no convergence proof for the case when a little deviation is not permitted, the programs run in [B39] converged to the optimum.

(12) Cabot's Method (see [B4])

The method is based upon the Tui cuts (see [B37]) and on the branch-and-bound technique.

(13) Konno's Method (see [B20])

This method is only for quadratic functions and is based on Konno's method for solving bilinear problems (see Method (20)).

A method for solving optimization problems with quasi-concave cost function (not necessarily quadratic) is:

(14) Majthay and Whinston's Method (see [B22])

This method is based on placing additional cuts to the constraint region and uses the fact that the optimal solution is located at an extreme point of the original constraint region. Therefore, the algorithm uses a marking method for differentiating the original constraints from those added by the algorithm.

It is also possible to solve quadratic programs with quasi-concave or concave cost functions by using Cabot and Francis' method (see Method (22)), or by using all the methods appearing in Section B.3.3.

B.3.3. General Methods for Solving Nonconvex Quadratic Programs

If the quadratic function has none of the properties mentioned in Sections B.3.1 and B.3.2, the quadratic program can be solved by using one of the general methods for solving nonconvex quadratic programs.

Since Problem 3.2 is a nonconvex quadratic program, the methods appearing in this section are more extensively explained; special attention is given to Ritter's algorithm, since it is the basic work on the subject.

(15) Majthay, Whinston and Coffman's Method (see [B23])

This method is almost identical to Ritter's method for finding a local minimum, with the only difference that Cottle and Mylander's presentation is used instead of the original Ritter presentation (see Method (19)).

Phase 3

As said before, in this phase a cut is added. This cut excludes from the constraint region all regions in which the cost function cannot achieve a smaller value than the value achieved in Phase 2. This is done by finding a global minimum to a new problem constructed from data of the final step of Phase 2. This new problem has only a capacity constraint $\underline{e}^T\underline{x} \leq \tau$ and can be easily solved for fixed τ. Then the solution to this problem is used for the construction of the cutting plane.

The algorithm was developed by Ritter (see [B27] and [B28]), but in [B9] Cottle and Mylander present it in a more elegant and understandable way.

In [B28] Ritter presents a proof of convergence to the global optimum in a finite number of steps, but in [B40] Zwart shows a counterexample and explains why Ritter's proof is not always true.

B.3.4 Methods for Solving Problems with Cost Functions having Special Properties

Some of the most important methods for solving special quadratic programs are:

(20) Methods for Bilinear Programs (see [B19])

A bilinear program is a special case of quadratic programming and can be defined as follows:

$$\min z(\underline{x},\underline{y}) = \underline{c}_1^T\underline{x} + \underline{c}_2^T\underline{y} + \underline{x}^T C\underline{y} ,$$

subject to

$$A_1\underline{x} = \underline{b}_1, \quad A_2\underline{y} = \underline{b}_2 , \qquad (B.7)$$

$$\underline{x} \geq 0 , \qquad \underline{y} \geq 0 .$$

One of the methods for solving this problem is Konno's method (see [B19]). This method is the basis for Konno's algorithm for solving concave quadratic programs (see Method (13).

(21) Method for Solving the Quasi-Concave Problem

$$\min z = (\underline{c}^T\underline{x}+\alpha)(\underline{d}^T\underline{x}+\beta)$$

$$\text{s.t.} \quad A\underline{x} = \underline{b} \quad , \qquad \underline{x} \geq 0 \quad , \tag{B.8}$$

where $(\underline{c}^T\underline{x}+\alpha)$ and $(\underline{d}^T\underline{x}+\beta)$ are strictly positive for all feasible solutions.

The method was developed by Swarup (see [B31], [B32] and [B33])

(22) Cabot and Francis' Method (see [B5] and [B24])

There are certain nonconvex quadratic programs that have an optimal solution in one of the extreme points of the convex set of feasible solutions (like those with a concave or quasi-concave performance index). These problems can be solved by Cabot and Francis' algorithm for ranking the extreme points.

In Chapter 5 it was proven that Problem 3.2 has the required property, therefore it will be explained in detail.

In this method the extreme points are ranked in increasing order of a linear function received from the quadratic problem (this is done by Murty's algorithm). In each step an upper and lower bound to the optimum are calculated for determining whether the end condition is satisfied or if it is necessary to continue with the algorithm.

Before presenting the algorithm, we present its theoretical background and Murty's algorithm.

The problem to be solved is:

Problem B.1 (denoted by P1)

$$\min f(\underline{x}) = \underline{c}^T\underline{x} + \underline{x}^T D\underline{x}$$

$$\text{s.t.} \qquad \underline{x} \in S$$

where

$$S = \{\underline{x}/A\underline{x} = \underline{b}, \quad \underline{x} \geq 0\} \ ,$$

knowing that it has characteristics that justify the extreme point assumption.

□ Problem B.1

If we denote by $\underline{d}_j$ the j-th column of D, and let

$$u_j = \min \underline{d}_j^T\underline{x} \qquad (B.9)$$

$$\text{s.t.} \quad \underline{x} \in S \ , \quad \forall j = 1,\ldots,n \ ,$$

we may state the boundary linear programming problem:

Problem B.2 (denoted by P2)

$$\min g(\underline{x}) = \sum_{j=1}^{n} (c_j+u_j)x_j = (\underline{c}^T+\underline{u}^T)\underline{x}$$

$$\text{s.t.} \qquad \underline{x} \in S \ .$$

□ Problem B.2

It can be easily proved that:

Proposition B.2

$$g(\underline{x}) \leq f(\underline{x}), \quad \forall x \in S$$

□ Proposition B.2

And as a consequence of this we can state that:

Corollary B.1

If $\underline{x}_o$ is an optimum solution to P2, then $f_\ell = g(\underline{x}_o)$ is a lower bound on f^* ($f^* = f(\underline{x}^*)$ is the optimal value of f).

□ Corollary B.1

We can also easily compute the upper bound $f_u = f(\underline{x}_o)$. With this upper and lower bound we conclude:

Corollary B.2

Given any upper bound f_u on f^*, denote by $\{\underline{x}_k\}$ the set of all extreme points $\underline{x}_k$ of P2, such that $g(\underline{x}_k) \leq f_u$. Then P1 has an optimum solution such that $\underline{x}^* \in \{\underline{x}_k\}$.

□ Corollary B.2

Murty's Algorithm

This algorithm was developed to solve the fixed charge problem:

$$\min_{\underline{x} \in S} \zeta(\underline{x}) = D(\underline{x}) + z(\underline{x})$$

where

$$S = \{\underline{x}/A\underline{x} = \underline{b}, \quad \underline{x} \geq 0\} ,$$

and

$$z(\underline{x}) = \sum_{j=1}^{n} c_j x_j, \quad D(\underline{x}) = \sum_{j=1}^{n} d_j (1-\delta_{o,x_j}) \qquad (B.10)$$

with

$$\delta_{o,x_j} = \begin{cases} 1 & \text{if } x_j = 0 \\ 0 & \text{if } x_j > 0 \end{cases} ,$$

and has two parts:

1. An algorithm for ranking the vertices of S in increasing order of $z(\underline{x})$.

2. An algorithm for the fixed charge problem, assuming that the vertices of S can be ranked in increasing order of the variable costs $z(\underline{x})$.

For our purpose we are interested in ranking the vertices of S in increasing order of

$$g(\underline{x}) = \sum_{j=1}^{n} (c_j + u_j) x_j ,$$

that is to say, in ranking the vertices of S in increasing order of a linear function; then, only the first part of Murty's method is relevant to us:

An Algorithm for Ranking the Vertices of S in Increasing Order of $g(\underline{x})$

(first part of Murty's algorithm)

Consider the linear program P2.

$$\min g(\underline{x}) = \sum_{j=1}^{n} (c_j + u_j) , \quad \text{s.t.} \quad \underline{x} \in S .$$

Assuming that a finite optimum exists, it is well known that there exists a vertex that is optimal. Consider any basic solution B, corresponding to any nonbasic variable $x_j \notin B$, let:

$\bar{c}_j^B \triangleq$ relative cost of coefficient of the nonbasic variable x_j corresponding to the basis B.

$\bar{\theta}_j^B \triangleq$ the value with which x_j enters the basis.

$T_j^B \triangleq$ the new basic feasible solution.

Then from the SIMPLEX algorithm follows that

$$g(T_j^B) = g(B) + \bar{\theta}_j^B \bar{c}_j^B ,$$

where T_j^B represents the adjacent extreme points to the extreme point represented by B.

Let $F(B)$ denote the set of all adjacent vertices of B with cost value not less than that of B; i.e.,

$$F(B) \triangleq \{(T_j^B : 0), \ \forall j / x_j \notin B, \ \bar{c}_j^B \geq 0\} . \qquad (B.12)$$

But, if B is a degenerate basic feasible solution, let V_B denote the vertex represented by it and let $B_1, \ldots, B_r$ be all the basic solutions representing the same vertex V_B. Then we should replace (B.12) by

$$F(V_B) = \bigcup_{p=1}^{r} \{(T_j^{B_p}:0),\ \forall j/x_j \notin B_p,\ \bar{c}_j^{B_p} \geq 0\} \quad . \qquad (B.13)$$

Proposition B.3

Every feasible basic solution can be reached by taking a path with nondecreasing cost (with $g(\underline{x})$ as the cost function) from $\underline{x}_o$ through the vertices of S (where $\underline{x}_o$ is the optimal solution of P2).

□ Proposition B.3

Denoting by $\underline{x}_i$ any specific basic solution, and taking $\{\underline{x}_o,\ldots,\underline{x}_{k-1}\}$ as the path with nondecreasing cost mentioned in Proposition B.3, we can state:

Proposition B.4

Suppose $\{\underline{x}_o,\ldots,\underline{x}_{k-1}\}$ are already known. Let us define

$$F_k = \bigcup_{i=o}^{k-1} F\{V_{\underline{x}_i}\} - \{\underline{x}_o,\ldots,\underline{x}_{k-1}\} \quad .$$

Then $\underline{x}_k$ is the minimal cost solution in F_k.

□ Proposition B.4

Based on the preceding facts, we can state the Cabot-Francis algorithm as follows (given in flow diagram form):

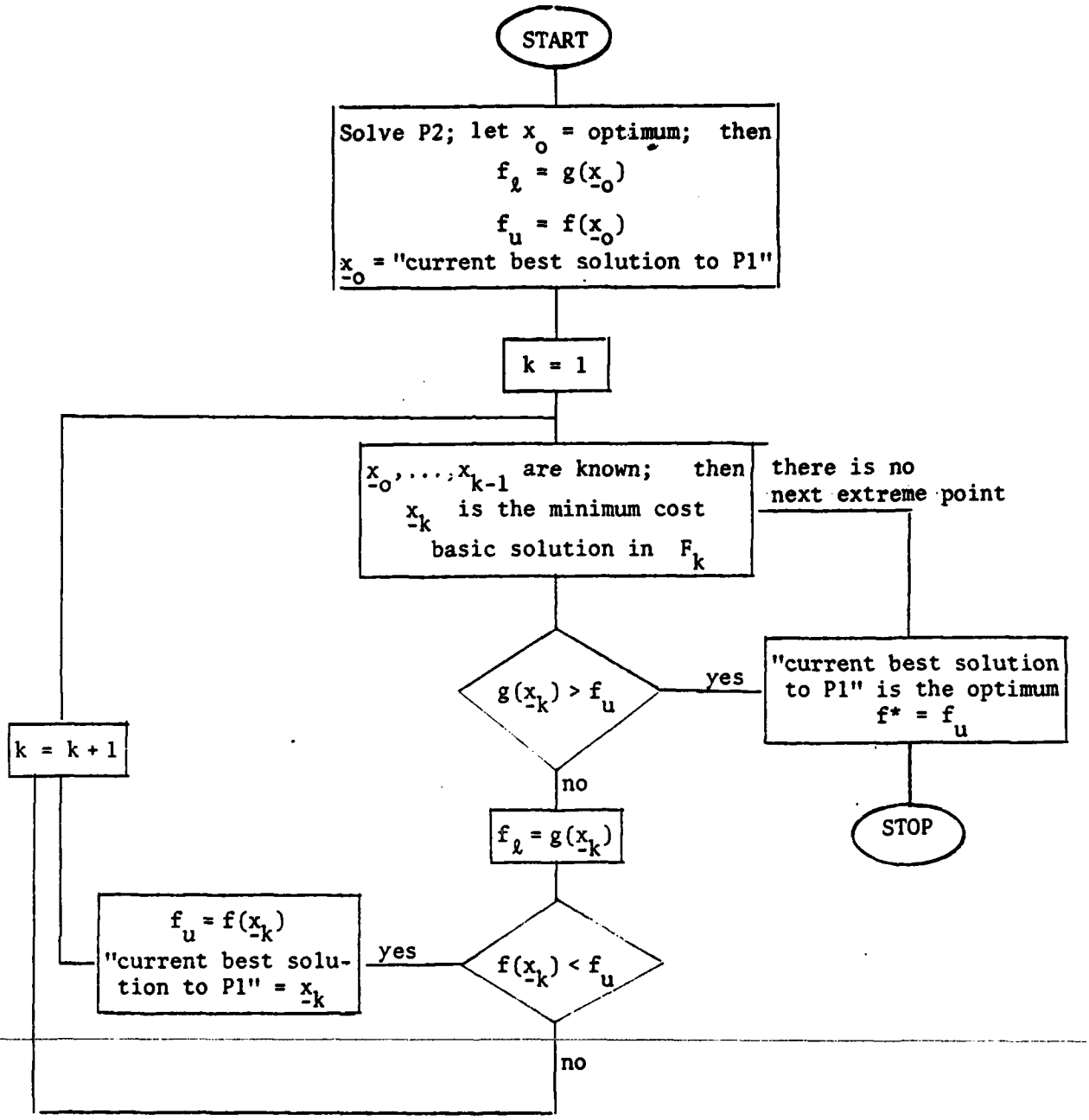


Figure B.1: The Cabot-Francis Algorithm

For a better understanding of the algorithm we will solve now a small example:

Example B.1

Find the minimum of

$$f(\underline{x}) = x_1 - x_2 - \frac{1}{2}x_1^2 + x_1x_2 + \frac{1}{2}x_2^2 \quad ,$$

subject to

$$\begin{aligned} 2x_1 + x_2 &\leq 4 \\ -x_1 + 3x_2 &\leq 3 \\ x_1 \geq 0, \quad x_2 &\geq 0 \end{aligned} \qquad \text{(B.14)}$$

The constraint region is represented in the following figure:

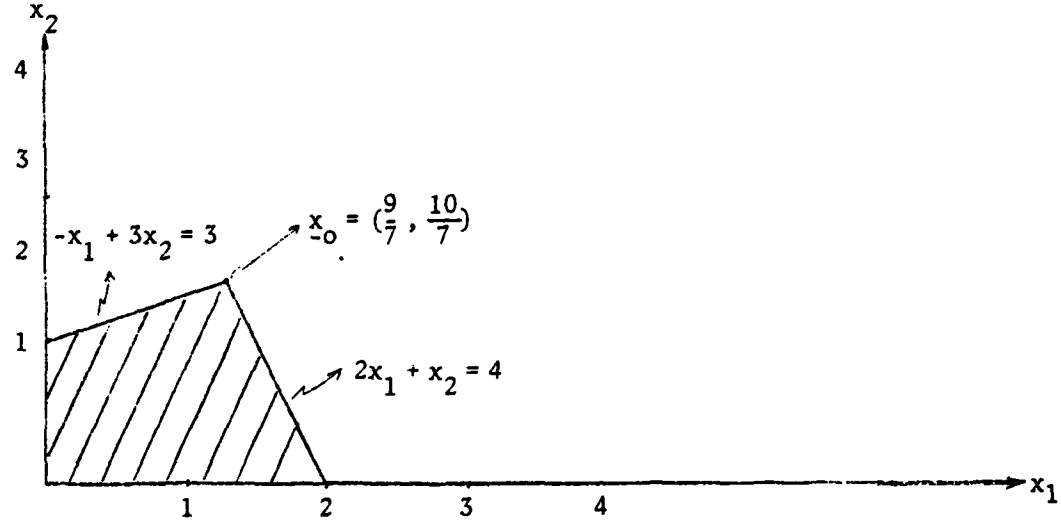


Figure B.2: Constraint region for Example B.1

For solving the problem we must express (B.14) in the standard form of P1, and this is done by adding two slack variables ($\underline{v}_1$ and $\underline{v}_2$). The resulting problem is:

Find the minimum of

$$f(\underline{x}) = (1,-1,0,0)\underline{x} + \underline{x}^T \begin{bmatrix} -1/2 & 1 & 0 & 0 \\ 0 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \underline{x} \Bigg\} , \qquad \text{(B.15)}$$

$$\text{s.t.} \quad \left.\begin{aligned} 2x_1 + x_2 + v_1 &= 4 \\ -x_1 + 3x_2 + v_2 &= 3 \\ \underline{x} &\geq 0 \end{aligned}\right\} S$$

where $\underline{x}^T = (x_1, x_2, v_1, v_2)$.

In order to obtain the boundary linear programming problem we must calculate u_j for $j = 1,\ldots,4$.

As said before,

$$u_j = \min_{\underline{x} \in S} \underline{d}_j^T \underline{x} \ , \quad \forall j = 1,\ldots,4 \ .$$

Hence:

$$u_1 = \min_{\underline{x} \in S} -\frac{1}{2} x_1 \quad \Rightarrow \quad u_1 = -1 \ ,$$

$$u_2 = \min_{\underline{x} \in S} x_1 + \frac{1}{2} x_2 \quad \Rightarrow \quad u_2 = 0 \ ,$$

$$u_3 = \min_{\underline{x} \in S} 0 \quad \Rightarrow \quad u_3 = 0 \ ,$$

$$u_4 = \min_{\underline{x} \in S} 0 \quad \Rightarrow \quad u_4 = 0 \ .$$

Therefore P2 is:

$$\min g(\underline{x}) = -x_2$$

$$\text{s.t.} \quad \underline{x} \in S \ .$$

From Figure B.2 follows that the minimum for $g(\underline{x})$ is achieved at:

$$\underline{x}_o = \left(\frac{9}{7}, \frac{10}{7}, 0, 0\right) \ .$$

Now we proceed with the algorithm proposed in Figure B.1.

1. We calculate f_ℓ and f_u.

$$f_\ell = g(\underline{x}_o) = -\frac{10}{7} \quad ,$$

$$f_u = f(\underline{x}_o) = \frac{185}{98} \quad .$$

Note that the SIMPLEX tableau representing the present situation (the tableau corresponding to the optimal solution of P2) is:

	x_1	x_2	v_1	v_2	z	
x_1	1	0	3/7	-1/7	0	9/7
x_2	0	1	1/7	2/7	0	10/7
	0	0	1/7	2/7	-1	10/7

2. From the tableau follows that the next-best extreme point is received by pivoting v_1 with x_1. The resulting tableau is:

	x_1	x_2	v_1	v_2	z	
v_1	7/3	0	1	-1/3	0	3
x_2	-1/3	1	0	1/3	0	1
	-1/3	0	0	1/3	-1	1

therefore, $g(\underline{x}_1) = -1$.

3. $-1 \not< \frac{185}{95}$ $(g(\underline{x}_1) \not< f_u)$,

hence $f_\ell = -1$

$f_{(\underline{x}_1)} = -1/2$.

4. $-1/2 < \frac{185}{95}$ $(f(\underline{x}_1) < f_u)$

hence

$f_u = -1/2$.

5. From the tableau follows that the next-best extreme point is received by pivoting v_2 with x_2. The resulting tableau is:

	x_1	x_2	v_1	v_2	z	
v_1	2	1	1	0	0	4
v_2	-1	3	0	1	0	3
	0	-1	0	0	-1	0

therefore, $g(\underline{x}_2) = 0$.

6. $0 > -\frac{1}{2}$ $(g(\underline{x}_2) > f_u)$

Hence, $\underline{x}_1$ is the optimal solution:

$$(x_1^*, x_2^*) = (0,1)$$
and
$$f(\underline{x}^*) = -1/2$$

□ Example B.1

(16) Candler and Townsley's Method (see [B6])

In this method a SIMPLEX-based technique is used for finding a local minimum to (B.1). At this point the cost function is represented as a function of the nonbasic variables and the partial derivatives of the cost function with respect to the nonbasic variables is calculated. These derivatives are the basis for the calculation of a cut that eliminates from consideration the local minimum, and other points with greater value of the cost function, but does not eliminate points with smaller cost function values. Then the steps of the algorithm are reapplied.

There exists no convergence proof for this method. In [B11] the proposed method was used for solving a simple example.

(17) Generalized Polars Method (see [B1] and [B3]

This relatively new approach is based upon the construction of the generalized polar of the Kuhn-Tucker polyhedron associated with the quadratic program (B.1). This generalized polar is a convex polyhedral cone whose interior contains no feasible solution better than the best known one. This and other properties are used to construct a polytope containing the feasible set and contained in the polar of the latter. This is done by starting with a simplex defined by a subset of the problem constraints, and successively activating other problem constraints, until the resulting polytope is contained in the polar of the feasible set. When this is achieved, the best complementary solution found in the process is optimal, or none exists.

(18) Tone's Method (see [B35])

This method is based on the solution of the generalized linear complementarity problem, received from the Kuhn-Tucker conditions generalized by the addition of a new variable.

The solution of the generalized linear complementarity problem is based on an algorithm for finding all extremal rays of a polyhedral convex cone with some complementarity conditions (see [B36]).

The convergence to a global minimum is guaranteed.

(19) Ritter's Algorithm (see [B9], [B27], [B28] and [B40])

This algorithm is one of the most important works on the subject. This fact led us to study it in depth.

The algorithm is iterative and each iteration is composed of the following three phases (the phases will be explained later):

Phase 1: Determines a vector that satisfies the constraints and expresses the objective function in terms of the independent (nonbasic) variables.

Phase 2: Determines a local minimum, with the extreme point that has been determined in Phase 1 as a starting point.

Phase 3: Constructs a cutting plane that excludes the previously located local minimum, without excluding the global minimum if the latter has not been found yet.

After the cutting plane has been placed, Phase 1 is reapplied to the augmented problem. Termination can occur in Phase 1, if no feasible points remain after placing the cutting plane; in Phase 2, with an indication that the objective function is not bounded below; or in Phase 3, if a weak sufficiency condition for a global minimum is satisfied, or with an indication that is not bounded below on the constraint set.

The problem to be solved is:

$$\min f(x) = \underline{c}^T\underline{x} + \frac{1}{2}\underline{x}^T D\underline{x} \quad ,$$

subject to

$$\left.\begin{array}{l} A\underline{x} \geq \underline{b} \\ \underline{x} \geq 0 \end{array}\right\} S \quad , \tag{B.4}$$

or, by writing the constraints more concisely:

$$\left.\begin{array}{l} \min f(\underline{x}) = \underline{c}^T\underline{x} + \frac{1}{2}\underline{x}^T D\underline{x} \\ \text{subject to} \\ G\underline{x} \geq \underline{h} \end{array}\right\} S \quad . \tag{B.5}$$

For its solution we use the following results:

Lemma B.1

For the set of all $\hat{\underline{h}}$ such that the Kuhn-Tucker conditions of (B.5) have a solution, f can be regarded as a function of $\hat{\underline{h}}$ and in this sense

$$\hat{\underline{\lambda}}^T = \nabla_{\hat{\underline{h}}} f \quad ,$$

$\hat{\underline{\lambda}}$ being the Lagrange multipliers that are greater than zero, and $\underline{h}$, G being the appropriate parts of $\underline{h}$ and G respectively.

□ Lemma B.1

Theorem B.2

If $(\underline{x}^*, \hat{\underline{\lambda}}^*)$ is a solution to the Kuhn-Tucker conditions of (B.5), then $\underline{x}^*$ is a local minimum of f on S iff f is convex on the set

$$\hat{L} = \{\underline{x} \in R^n / \hat{G}\underline{x} = \hat{\underline{h}}\} \quad .$$

□ Theorem B.2

Proposition B.1

If $f(\underline{x}^*)$ is a local minimum of (B.5), and $\min_{\underline{x} \in S}\{\underline{x}^T D\underline{x} / G_o\underline{x} \geq 0\} \geq 0$ then, $f(\underline{x}^*)$ is a global minimum of f on S.

Where G_o is the part of G satisfying the constraints as equalities, i.e. $G_o \underline{x}^* = \underline{h}_o$.

□ Proposition B.1

Note that Proposition B.1 is the previously mentioned weak sufficiency condition for a global minimum.

Using the above results we proceed to explain the three phases of the algorithm:

Phase 1

We transform the inequality constraints to equality constraints by adding slack variables ($\underline{v}$).

In order to find a feasible solution we add artificial variables r_i, and by using the SIMPLEX method we minimize $\sum_{i=1}^{n} r_i$. Termination can occur with a feasible point (all $r_i = 0$), or with an indication that there is no feasible point.

After finding the feasible point, we write the problem in such a way that f depends only on the nonbasic variables. It must be noted that after doing that, it may happen that f will depend not only on elements of $\underline{x}$, but on elements of $\underline{v}$ as well.

Now we redefine $\underline{x}$ in such a way that $\underline{x} = 0$ is a feasible point.

Phase 2

As said before, in this phase a local minimum is found.

The previous application of Phase 1 guaranties that $\underline{x} = 0$ is a feasible point. The algorithm works by maintaining a solution to the Kuhn-Tucker conditions, which yields to a local minimum for (B.5) augmented with the capacity constraint:

$$\underline{e}^T \underline{x} \leq \tau$$

with

$$e_i \geq 0 \qquad \text{(B.6)}$$

and

$$e_i \geq 0 \quad \text{if} \quad c_i \leq 0 \ .$$

With the parameter $\tau = 0$, the point $\underline{x} = 0$ is obviously a local minimum for the augmented problem, since no other point is feasible. At this point, the capacity constraint is binding and has a positive multiplier. Then, the capacity constraint is relaxed by an increase of the parameter and the Kuhn-Tucker conditions for the augmented problem are used to compute a local minimum $x = x(\tau)$ for this prohlem.

However, there are conditions under which it is impossible to increase τ so that $\underline{x}(\tau)$ remains a local minimum of the augmented problem. Depending on how the situation arises, it will be necessary either to specify that certain nonnegative constraints $(x_i \geq 0)$ be treated as equality constraints $(x_i = 0)$, or to change the capacity constraints.

Lemma B.1 shows that as τ is being increased, the objective function is being decreased as long as the capacity constraint has a positive multiplier. After a finite number of iterations, the algorithm either terminates with an indication that the objective function has no finite lower bound on the feasible region, or the multiplier of the capacity constraint reaches the value zero. When this occurs, we have a local minimum to the problem without the capacity constraint, but we may still have the imposed condition that some of the variables be held at the value zero. If there are no such restrictions, the point at hand is a local minimum to (B.5) and Phase 3 is to be executed. However, if we have a local minimum to the problem with the additional condition that some of the variables be equal to zero, then a new capacity constraint is to replace the one just dropped and Phase 2 must be reapplied.

Appendix C

Computation Program for Solving the Open-Loop Optimal Dynamic Routing Problem for Single Destination Networks with Unity Weightings in the Cost Functional

C.1 Introduction

The program presented in this appendix is an implementation of the algorithm proposed in Chapter 4. It permits to solve the open-loop optimal dynamic routing problem for single-destination networks with unity weightings in the cost functional, by using linear programs only (see Section 4.3).

The proposed computation program is written in FORTRAN and makes use of the program LA01B, from the HARWELL Library, for solving linear programs.

C.2 The Program Input

M - number of links in the network

N - number of nodes in the network

B - bidimensional array defining the network topology ($\dot{\underline{x}} = B\underline{u}$). Its dimension is (N,M).

CA - M-dimension vector representing the capacity of the links

X□ - initial condition vector of dimension N

All this data is read according to a FORMAT appearing in the first part of the program (see Section C.6).

IA - maximal number of rows in the A-matrix, i.e. α (see Section C.4).

This data is inserted directly into the program (see Section C.6).

C.3 The Program Output

The program prints first the problem parameters, i.e., the capacity vector, the B-matrix and the initial condition vector (see Section C.6). Then it prints the optimal solution, i.e., the optimal flow in each link (the vector $\underline{u}$), and the time period this control is applied. It is also possible to obtain error messages from the linear programs and intermediate results from the algorithm, by using IPR = 1 instead of 0.

C.4 Dimensioned Variables

A - Bidimensional array defining the constraint region of the linear program to be solved (see Section C.5). Its dimension is (α,β) where:

$$\alpha = 2N(L'+1) + M(2L'+2) + 2L' + 1 ,$$

$$\beta = M(2L'+2) + L'N + 3L' + 3 ,$$

L' being the maximal value L may achieve ($L = 0,1,3,7,\ldots,L'$). Remember that the number of switches in Operation 2 is $2L+1$ (see Figure 4.16).

BD - Vector defining the constraint region of the linear program to be solved. Its dimension is α.

C - Vector of the cost weightings. Its dimension is β.

IND - Vector required by LA01B. Its dimension is γ where:

$$\gamma = \alpha + N(L'+1) + M(2L'+2) .$$

WK - Vector required by LA01B. Its dimension is $\gamma(\alpha+3)$.

TAU - Vector of the time intervals $\tau_i = t_i - t_{i-1}$, $i = 1,\ldots,2\ell+2$ where $t_{2\ell+2} = t_f^*$. Its dimension is $2L' + 2$.

X - Vector of the variables that are to be optimized in the linear program. Its dimension is β.

Z - Vector of dimension $M(2L'+2)$.

SX - Vector of dimension L'.

SP - Vector of dimension L'.

C.5 Explanation of the Problem

In the first part of the program the parameters of the problem to be solved are read and written. In the second part the constraint region $AX \leq BD$ is built, where

$$A = \begin{pmatrix} I - CA \\ \hline B \quad 0 \end{pmatrix},$$

$$BD = \begin{pmatrix} Q \\ \hline X\square \end{pmatrix},$$

$$X = \begin{pmatrix} z \\ \hline t \end{pmatrix}.$$

Note that the constraints over the double line are constraints of the type $\underline{a}^T\underline{x} \leq b$ (the LA01B program adds slack variables and transforms them into equality constraints), and a cost function C^TX is built where

$$C = \begin{pmatrix} 0 \\ \hline I \end{pmatrix}.$$

It is easy to verify that the linear program defined above corresponds to Operation 1 of the algorithm (see (4.22)).

After solving this linear program, the non-switch-optimal-solution, the final time and the cost function value are stored in memory.

In the third part of the program, the constraint region and cost function corresponding to Operation 2 are built (see (4.23)), and the resulting linear program is solved. The corresponding A, BD, C and X are:

$$
A = \begin{array}{|c|c|c|c|l}
\multicolumn{1}{c}{M(2L+2)} & \multicolumn{1}{c}{LN} & \multicolumn{1}{c}{2L+2} & \multicolumn{1}{c}{L+1} & \\
\hline
\begin{matrix} -B & & \\ & -B & \\ & & -B \end{matrix} & \begin{matrix} I & & \\ -I & I & \\ & -I & I \\ & & -I \end{matrix} & 0 & 0 & N(L+1) \\
\hline
\begin{matrix} I & & & \\ & I & & \\ & & I & \\ & & & I \end{matrix} & 0 & \begin{matrix} -CA & & & \\ & -CA & & \\ & & -CA & \\ & & & -CA \end{matrix} & 0 & M(2L+2) \\
\hline
\begin{matrix} -B & -B & & \\ & -B & -B & \\ & & -B & -B \end{matrix} & \begin{matrix} I & & \\ -I & I & \\ & -I & I \end{matrix} & 0 & 0 & N(L+1) \\
\hline
0 & \begin{matrix} \underline{1}^T & \\ & \underline{1}^T \end{matrix} & 0 & 0 & L \\
\hline
0 & 0 & \begin{matrix} 1 & 1 & \\ & 1 & 1 \end{matrix} & \begin{matrix} 1 \\ 1 \\ 1 \end{matrix} & L+1
\end{array}
$$

$$BD = \begin{bmatrix} \underline{X}\square \\ \underline{0} \\ \hline \underline{0} \\ \hline \underline{X}\square \\ \underline{0} \\ \hline SX(1)' \\ \vdots \\ SX(L)' \\ \hline TAU(1)' \\ \vdots \\ TAU(L+1)' \end{bmatrix} \begin{matrix} \} N(L+1) \\ \} M(2L+2) \\ \} N(L+1) \\ \} L \\ \} L+1 \end{matrix} \qquad X = \begin{bmatrix} \underline{z}_1 \\ \vdots \\ \underline{z}_{2L+2} \\ \hline \underline{x}(t_1') \\ \vdots \\ \underline{x}(t_L') \\ \hline TAU(1) \\ \vdots \\ TAU(2L+2) \end{bmatrix} \begin{matrix} \} M(2L+2) \\ \} LN \\ \} 2L+2 \end{matrix}$$

$$C^T = \underbrace{\left[\underline{0}^T, \left(\frac{-\underline{1}^T BTAU(1)'}{2}\right)^T, \underline{0}^T, \ldots, \left(\frac{-\underline{1}^T BTAU(L+1)'}{2}\right)^T\right]}_{M(2L+2)} \underbrace{\vdots \underline{0}^T \vdots}_{LN} \underbrace{\frac{SX\square - SX(1)'}{2}, 0, \frac{SX(1)' - SX(2)'}{2}}_{2L+2},$$

$$\ldots, \underbrace{\frac{SX(L)' - SX(L+1)'}{2}}, \underbrace{0 \vdots \underline{0}^T}_{L+1}\Bigg],$$

where $SX\square = \sum_{i=1}^{n} x\square_i$, $SX(i)' = \sum_{j=1}^{n} x_j'(t_i')$. The "prime" indicates that the respective value is known from the previous step of the algorithm. As said before, the constraints over the double line are inequality constraints of the type $\underline{a}^T \underline{x} \leq b$.

In the last part of the program, the optimal cost received from the last application of Operation 2 is compared with the stored value. If there is no decrease in the cost value, the optimal solution (i.e., the stored solution) is printed and the program stops; but if there is a decrease in the optimal cost, L is increased (L = 2L+1), and the last solution and cost value are stored instead of the previously stored solutions, and Operation 2 is reapplied. If there is some TAU(i)' equal to zero, the program elimin-

ates the appropriate elements from A, BD, C and X, and in this way further switches in this time interval are not permitted.

C.6 The Program

```
      COMMON /LAU10/EPS,MAXINV,NMAS1S,LP,LPD,TOL
      DIMENSION B(2,7),CA(7),XO(3),A(67,77),BD(67),C(77),IND(155)
     *,WK(13550),TAU(6),X(77),Z(56),SX(3),SP(3)
      IA=67
C*****************************************************************
C        DEFINITION OF THE PROBLEM TO BE SOLVED
C*****************************************************************
      IPR=0
      MAXINV=10
      LP=6
      LPD=0
      TOL=0.0001
      READ(5,62)EPS
   62 FORMAT(E4.0)
      READ(5,63)N
   63 FORMAT(30F2.0)
      READ(5,64)CA
   64 FORMAT(12F5.3)
      WRITE(6,61)CA
   61 FORMAT(1X,'THE CAPACITY VECTOR IS:',///,12(F6.3,1X),/,1X,12(F6.3,1
     *X))
      WRITE(6,52)
   52 FORMAT(/////,1X,'THE B MATRIX IS:')
      DO 53 I=1,N
      WRITE(6,54)(B(I,J),J=1,M)
   54 FORMAT(//,2X,24(F4.1))
      READ(5,65)XO
   65 FORMAT(15F4.2)
      WRITE(6,55)XO
   55 FORMAT(///,1X,'THE INITIAL CONDITION IS:',//,1X,15(F5.2,1X),/,1X,1
     *5(F5.2,1X))
      KL=0
C*****************************************************************
C  OPERATION 1 ; THE ZERO-SWITCH-OPTIMAL-SOLUTION,CALCULATION OF TF
C*****************************************************************
      NF=M+N
      NC=M+1
      DO 101 I=1,NF
      DO 101 J=1,NC
  101 A(I,J)=0
      DO 1 I=1,N
      K1=M+I
      A(K1,NC)=0
      BD(K1)=XO(I)
      DO 1 J=1,M
      A(K1,J)=B(I,J)
      IF(I-1)4,4,1
    4 A(J,J)=1
      A(J,NC)=-CA(J)
      BD(J)=0
      C(J)=0
    1 CONTINUE
      C(NC)=1
      NOASIS=0
      IPR=0
      CALL LAU10(NC,NF,A,BD,C,X,F,IA,IPR,IND,WK,IER)
      TAU(1)=X(NC)
      SYC=0
      DO 3 I=1,M
    3 Z(I)=X(I)
      DO 41 I=1,N
   41 SYC=SYC+XO(I)
      SP1=(SYC+X(NC))/2
      SX(1)=0
      WRITE(6,60) TAU(1)
   60 FORMAT(///,1X,'THE TOTAL TIME IS:',E13.7)
C*****************************************************************
C            OPERATION 2
C*****************************************************************
C
C CONSTRUCTION OF THE CONSTRAINT REGION ***************************
C
      L=0
  100 I=L*N-KL*N
      I7=2*L+2-2*KL
      K7=N*(L+1)-KL*N
      K8=L+1-KL
      K1=L+1-KL
```

```
      K7=N*17
      K4=K7+16+17+NP
      K5=2*K2+K7+L+NB-KL
      N10=2*K2+K7+L-KL
      N11=K7+16+17
      K1=K2+K7
      I4=K7+16
      K6=K1+K2
      DO 4 I=1,K5
      B(I)=0
      DO 4 J=1,K4
    4 A(I,J)=0
      DO 5 K=1,K5
      M1=2**(K-1)
      M2=N*(K-1)
      DO 5 I=1,M
      IF(K-1)12,12,13
   12 AO(I)=AO(I)
      J1=K1+I
      AO(J1)=AO(I)
   13 M3=M2+I
      M4=M3+K1
      DO 5 J=1,M
      J1=I1+J
      A(M3,M4)=-B(I,J)
      A(M4,M3)=-B(I,J)
      M5=M3+M
    5 A(M5,M4)=-B(I,J)
      DO 6 I=1,N7
      K=K6+I
    6 A(K,I)=1
      DO 7 I=1,17
      I5=I4+I
      DO 7 J=1,M
      I6=K2+M*(I-1)+J
    7 A(I6,I5)=-CA(J)
      IF(L)9,9,8
    8 DO 10 I=1,I2
      J8=K7+I
      A(I,J8)=1
      I9=N+I
      A(I9,J8)=-1
      J9=K1+I
      J7=J9+N
      A(J9,J8)=1
   10 A(J7,J8)=-1
      DO 11 I=1,L
      JP=K6+I
      B(JP)=SX(I)
      DO 11 J=1,N
      J9=K7+(I-1)*N+J
   11 A(JP,J9)=1
    9 DO 121 I=1,NC
      NN=N104+I
      B(NN)=TAU(I)
      MM=N11+I
      A(N9,MM)=1
      DO 121 J=1,2
      N12=I44+(I-1)*2+J
  121 A(N9,N12)=1
C
C*****DEFINITION OF THE PERFORMANCE INDEX  ***********************
C
      DO 20 I=1,K4
   20 C(I)=0
      DO 27 K=1,K6
      N3=2**K
      N4=2**K-1+I4
      IF(K-1)24,24,25
   24 C(N4)=(SXU-SX(1))/2
      GO TO 26
   25 IF(K-K6)26,27,27
   26 N8=K-1
      C(N4)=(SX(N8)-SX(K))/2
      GO TO 28
   27 N8=K-1
      C(N4)=SX(N8)/2
   28 DO 22 I=1,M
      SH=0
      N1=(N3-1)*M+I
      DO 21 J=1,N
   21 SH=SH+B(J,I)
   22 C(N1)=-SH*TAU(K)/2
      EPS=0
      NBASIS=0
      CALL LAU1(K4,K5,K1,A,HU,C,X,F,IA,IP,IND,WK,IER)
```

```
C     REVISION OF THE END CONDITION ; WRITTING OF THE SOLUTION VECTOR
C
      PIN=F
      IF(L)90,80,81
   81 LP=L-KL
      DO 23 I=1,LP
   23 PIN=PIN+SX(I)*TAU(I)
   30 IF(P1-PIN-TOL)30,30,31
   31 P1=PIN
      ML=0
      DO 710 I=1,17
      J=14+I
      IF(X(J)-TOL)711,711,712
  711 KL=KL+1
      GO TO 710
  712 ML=ML+1
      TAU(ML)=X(J)
      DO 714 K=1,M
      LL=(I-1)*M+K
      MK=(ML-1)*M+K
  714 Z(MK)=X(LL)
  710 CONTINUE
      IF(L)34,34,35
   35 DO 71 I=1,LP
   71 SP(I)=SX(I)
   34 SX(1)=1.0
      DO 36 J=1,M
      SF=0
      DO 37 I=1,N
   37 SF=SF+L(I,J)
   36 SX(1)=SX(1)+SF*X(J)
      IF(L)38,38,70
   70 DO 72 I=1,LP
      L3=2*I+1
      SA(L3)=SP(I)
      L4=2*I
      SX(L4)=SP(I)
      DO 72 J=1,M
      SU=0
      DO 73 II=1,N
   73 SU=SU+L(II,J)
      L5=(L3-1)*M+J
   72 SX(L3)=SX(L3)+SU*X(L5)
   38 L=2*L+1
      IF(KL)100,100,716
  716 MP=1
      DO 718 I=1,L
      J=I-1
      IF(I-1)720,720,721
  720 IF(SX0-SX(1)-TOL)718,718,717
  721 IF(SX(J)-SX(I)-TOL)718,718,717
  717 SP(MP)=SA(I)
      MP=MP+1
  718 CONTINUE
      K=L-KL
      DO 719 I=1,K
  719 SX(I)=SP(I)
      GO TO 100
   90 IF(L)90,96,91
   90 WRITE(6,93)TAU(1)
   93 FORMAT(///////,2X,'NO SWITCHS IN THE OPTIMAL SOLUTION',//,7X,'TAU(1
     #)=',E13.7)
      DO 92 I=1,M
      Z(I)=Z(I)/TAU(1)
   92 WRITE(6,94)I,Z(I)
   94 FORMAT(///,4X,'U(',I2,')=',E13.7)
      GO TO 700
   91 LM=L+1-KL
      MS=0
      DO 95 K=1,LM
      IF(TAU(K)-TOL)95,99,99
   99 MS=MS+1
      WRITE(6,96)MS,TAU(K)
   96 FORMAT(//////,7X,'TAU(',I2,')=',E13.7)
      DO 95 J=1,M
      LS=(K-1)*M+J
      Z(LS)=Z(LP)/TAU(K)
      WRITE(6,97)J,Z(LS)
   95 CONTINUE
   97 FORMAT(///,4X,'U(',I2,')=',E13.7)
  700 STOP
      END
```

References

1. General

[G1] Dantzig, G.B., Linear Programming and Extensions, Princeton University Press, Princeton, N.J., 1963.

[G2] Jodorkovsky, M., and Segall, A., "A Maximal Flow Approach to Dynamic Routing in Communication Networks," EE Pub. No. 358, Technion - Israel Institute of Technology, Aug. 1979.

[G3] Luenberger, D.G., Introduction to Linear and Nonlinear Programming, Addison-Wesley, 1973.

[G4] Moss, F.H., "The Application of Optimal Control Theory to Dynamic Routing in Data Communication Networks," Ph.D. Dissertation, Massachusetts Institute of Technology, Cambridge, MA, 1976.

[G5] Moss, F.H., and Segall, A., "An Optimal Control Approach to Dynamic Routing in Data Communication Networks, Part I: Principles," EE Pub. No. 312, Technion - Israel Institute of Technology, September 1977.

[G6] Moss, F.H., and Segall, A., "An Optimal Control Approach to Dynamic Routing in Data Communication Networks, Part II: Geometrical Interpretation," EE Pub. No. 319, Technion - Israel Institute of Technology, January 1978.

[G7] Segall, A., "The Modelling of Adaptive Routing in Data Communication Networks," IEEE Trans. on Comm., COM-25, No. 1, 85-95, January 1977.

[G8] Simmonard, M., Linear Programming, Prentice-Hall, 1966.

2. Properties of Functions

[A1] Arrow, K.J., and Enthoven, A.C., "Quasi-Concave Programming", Econometrica, 29 (1961), 779-800.

[A2] Avriel, M., Nonlinear Programming: Analysis and Methods, Prentice-Hall, 1976.

[A3] Cottle, R.W., and Ferland, J.A., "On Pseudo-Convex Functions of Non-negative Variables", Technical Report No. 70-9, Operations Research House, Stanford University, July 1970.

[A4] Cottle, R.W., "On the Convexity of Quadratic Forms over Convex Sets", Operations Res., 15 (1967), 170-172.

[A5] Debreu, G., "Definitive and Semidefinite Quadratic Forms", Econometrica, 20 (1952), 295-300.

[A6] Farebrother, R.W., "Necessary and Sufficient Conditions for a Quadratic Form to be Positive whenever a Set of Homogeneous Linear Constraints is Satisfied", Linear Algebra Appl., 16 (1977), 39-42.

[A7] Ferland, J.A., "Quasi-Convex and Pseudo-Convex Functions on Solid Convex Sets", Technical Report 71-4, Operations Research House, Stanford University, April 1971.

[A8] de Finetti, B., "Sulle Stratificazioni Convesse", Ann. Math. Pura Appl., (4) 30 (1949), 173-183.

[A9] Mangasarian, O.L., "Pseudo-Convex Functions", J. SIAM Control, 3 (1965), 281-290.

[A10] Mangasarian, O.L., Nonlinear Programming, McGraw-Hill, New York, 1969.

[A11] Martos, B., "Quadratic Programming with a Quasi-Convex Objective Function", Operations Research, 19 (1971), 82-97.

[A12] Martos, B., "Subdefinite Matrices and Quadratic Forms", SIAM J. Appl. Math., 17 (1969), 1215-1223.

[A13] Nikaido, H., "On von Neuman's Minimax Theorem", Pacific J. Math., 9 (1967), 115-119.

[A14] Ponstein, J., "Seven Kind of Convexity", SIAM Review, 9 (1967), 115-119.

[A15] Schaible, S., "Quasi-Convex Optimization in General Real Linear Spaces", Zeitschrift für Operations Research, 16 (1972) 205-213.

[A16] Schaible, S., "Quasi-Convex, Strictly Quasi-Convex and Pseudo-Concave Functions", Methods of Operations Research, Vol. 17, Meisenheim, West Germany, 308-316, 1973.

[A17] Schaible, S., "Quasi-Concavity and Pseudo-Concavity of Cubic Functions", Mathematical Programming, 5 (1973) 243-247.

[A18] Schaible, S., "Second Order Characterizations of Pseudo-Convex Quadratic Functions", Technical Report No. 75-29, Operations Research House, Stanford University, November 1975.

[A19] Tui, H., "Sur des inégalités linéaires", Colloquium Mathematicum, 13 (1964), 107-123.

3. Quadratic Programming

[B1] Balas, E., "Nonconvex Quadratic Programming via Generalized Polars", SIAM J. Appl. Math., 28 (1975), 335-349.

[B2] Beale, E.M.L., "On Quadratic Programming", Naval Res. Log. Quart., 6 (1959), 227-243.

[B3] Burdet, C.A., "Polaroids: A New Tool in Non-Convex and in Integer Programming", Naval Res. Log. Quart., Vol. 20, 1973.

[B4] Cabot, A.V., "Variation on a Cutting Plane Method for Solving Concave Minimization Problems with Linear Constraints", Naval Res. Log. Quart, 21 (1974), 265-274.

[B5] Cabot, A.V., and Francis, "Solving Certain Nonconvex Quadratic Minimization Problems by Ranking the Extreme Points", Operations Research, Vol. 18, No. 1 (1970).

[B6] Candler, W., and Townsley, R.J., "The Maximization of a Quadratic Function Subject to Linear Inequalities", Management Sci., 10 (1964), 515-523.

[B7] Carrillo, M.J., "A Relaxation Algorithm for the Minimization of a Quasi-Concave Function on a Convex Polyhedron", Mathematical Programming, 13 (1977), 69-80.

[B8] Cottle, R.W., and Dantzig, G.B., "Complementary Pivot Theory of Mathematical Programming", J. Linear Alg. Appl., 1 (1968), 103-125.

[B9] Cottle, R.W., and Mylander, W.C., "Ritter's Cutting Plane Method for Nonconvex Quadratic Programming", Technical Report No. 69-11, Operations Research House, Stanford University, July 1969.

[B10] Elzinga, J., and Moore, T.G., "A Central Cutting Plane Algorithm for the Convex Programming Problem", Mathematical Programming, 8 (1975), 134-145.

[B11] Evans, D.A., "A Simple Worked Example of the Maximization of a Quadratic Function Subject to Linear Inequalities", Technical Discussion Paper No. 1, Dept. of Agricultural Economics, Massey University College, New Zealand, 1963.

[B12] Frank, M., and Wolfe, P., "An Algorithm for Quadratic Programming", Naval Res. Log. Quart., 3 (1956), 95-110.

[B13] Hadley, G., Nonlinear and Dynamic Programming, Addison-Wesley, 1967.

[B14] Hefley, G.L., "A Comparison of Two Quadratic Programming Algorithms", Technical Report No. 41, Systems Research Center, University of Florida, 1970.

[B15] Hildreth, C., "A Quadratic Programming Procedure", Naval Res. Log. Quart., 14 (1957), 79-85.

[B16] Houthakker, H.S., "The Capacity Method of Quadratic Programming", Econometrica, 28 (1960), 62-87.

[B17] Hu, T.C., "Minimizing a Concave Function in a Convex Polytope", Technical Summary Report No. 1011, Mathematics Research Center, U.S. Army, 1969.

[B18] Keller, E.L., "Quadratic Optimization and Linear Complementarity", Doctoral Dissertation, University of Michigan, Ann Arbor, 1969.

[B19] Konno, H., "A Cutting Plane Algorithm for Solving Bilinear Programs", Mathematical Programming, 11 (1976), 14-27.

[B20] Konno, H., "Maximization of Convex Quadratic Function Under Linear Constraints", Mathematical Programming, 11 (1976), 117-127.

[B21] Lemke, C.E., "A Method for Solution of Quadratic Programs", Management Science, 8 (1962), 442-453.

[B22] Majthay, A., and Whinston, A., "Quasi-Concave Minimization Subject to Linear Constraints", Discrete Mathematics, 9 (1974), 35-39.

[B23] Majthay, A., Whinston, A., and Coffman, J., "Local Optimization for Nonconvex Quadratic Programming", Naval Res. Log. Quart., 21 (1974), 465-490.

[B24] Murty, K., "Solving the Fixed-Charge Problem by Ranking the Extreme Points", Operations Research, 16 (1968), 268-279.

[B25] Mylander, W.C., "Finite Algorithms for Solving Quasi-Convex Quadratic Programs", Operations Research, 20 (1972), 167-173.

[B26] Mylander, W.C., "Nonconvex Quadratic Programming by a Modification of Lemke's Method", Technical Paper RAC-TP-414, Research Analysis Corp., McLean, Virginia, 1971.

[B27] Ritter, K., "Stationary Points of Quadratic Maximum Problems", Z. Wahrscheinlichkeitstheorie verw. Geb., 4 (1965), 149-158.

[B28] Ritter, K., "A method for Solving Maximum Problems with a Nonconcave Quadratic Objective Function", Z. Wahrscheinlichkeitstheorie verw. Geb., 4 (1966), 340-351.

[B29] Shetty, C.M., "A Simplified Procedure for Quadratic Programming", Operations Research, 11 (1963), 248-260.

[B30] Simmons, D.M., Nonlinear Programming for Operations Research, Prentice-Hall, N.J., 1975.

[B31] Swarup, K., "Programming with Indefinite Quadratic Function with Linear Constraints", Cahiers du Centre d'Etudes de Recherche Opérationelle, 8 (1966).

[B32] Swarup, K., "Indefinite Quadratic Programming", Cahiers du Centre d'Etudes de Recherche Opérationelle, 8 (1966).

[B33] Swarup, K., "Quadratic Programming", Cahiers du Centre d'Etudes de Recherche Opérationelle, 8 (1966).

[B34] Theil, H., and van de Panne, C., "Quadratic Programming as an Extension of Classical Quadratic Maximization", Management Science, 7 (1960), 1-20.

[B35] Tone, K., "A Method for Nonconvex Quadratic Programming", Keio Engineering Reports, 27 (1974), 113-125.

[B36] Tone, K., "An Algorithm for Finding All Extremal Rays of Polyhedral Convex Cones with some Complementarity Conditions", to appear in J. of the Operations Research Society of Japan.

[B37] Tui, H., "Concave Programming under Linear Constraints", Dokl. Akad. Naul. SSR., 159 (1964), 1437-1440.

[B38] Wolfe, P., "The Simplex Method for Quadratic Programming", Econometrica, 27 (1959), 382-398.

[B39] Zwart, P.B., "Global Maximization of a Convex Function with Linear Inequality Constraints", Operations Research, 22 (1974), 602-609.

[B40] Zwart, P.B., "Nonlinear Programming: Counterexamples to Global Optimization Algorithms by Ritter and Tui", Operations Research, 21 (1973), 1260-1266.